Delegation of Computation with Verification Outsourcing: Curious Verifiers

Gang Xu
Department of Electrical and Computer Engineering
Iowa State University
Ames, Iowa 50011
gxu@iastate.edu

George Amariucai
Department of Electrical and Computer Engineering
Iowa State University
Ames, Iowa 50011
gamari@iastate.edu

Yong Guan
Department of Electrical and Computer Engineering
Iowa State University
Ames, Iowa 50011
guan@iastate.edu

ABSTRACT

In the Cloud Computing paradigm, a user often reduces financial, personnel, and computational burdens by outsourcing computation and other IT services to a professional service provider. However, to be able to assure the correctness of the result, the user still needs to perform the verification himself. Such verification may be tedious and expensive. Consequently, users are likely to outsource (again) the verification workload to a third party. Other scenarios such as auditing and arbitrating may also require the use of third-party verification. Outsourcing verification will introduce new security challenges. One such challenge is to protect the computational task and the results from the untrusted third party verifier. In this work, we address this problem by proposing an efficient verification outsourcing scheme. To our knowledge, this is the first solution to the verification outsourcing problem. We show that, without using expensive fully-homomorphic encryption, an honest-but-curious third party can help to verify the result of an outsourced computational task without having to learn either the computational task or the result thereof. We have implemented our design by combining a novel commitment protocol and an additive-homomorphic encryption in the argument system model. The total cost of the verification in our design is less than the verifier's cost in the state-of-the-art argument systems that rely only on standard cryptographic assumptions.

Categories and Subject Descriptors

F.1.2 [**Theory of Computation**]: Modes of Computation; F.2.0 [**Theory of Computation**]: Analysis of Algorithms and Problem Complexity—*General*; K.6.5 [**Management of Computing and Information Systems**]: Security and Protection

General Terms

Algorithms, Security, Theory

PODC'13, July 22–24, 2013, Montréal, Québec, Canada.

Keywords

Delegation of verification, argument systems, PCPs, cloud computing, delegating computation, privacy

1. INTRODUCTION

Cloud Computing represents a new trend in modern computing. Since computation can be purchased as a service, companies and individual users can cut down their computing assets and outsource any burdensome computational workload. In addition to savings in computing infrastructure, the Cloud may also provide expert technical consulting. But while outsourcing computation provides appealing benefits, one must fully consider a critical security issue: there is no guarantee on the correctness of the results. That is, the Cloud servers should be considered error-prone and may or may not be fully trustable. Thus an immediate need for *result assurance* naturally arises.

This need motivates a growing body of research on verification of outsourced computation. Some recent works focus on specific problems and exploit properties of these problems for efficient verification [11] [13] [22] [31] [33] [44] [45] [46] [47] [4] [24] [40]. Others strive for verifying the result of general computation, not limited to a specific computational task. Extending classical proof systems, interactive proof (IP) systems [30] and probabilistically checkable proof (PCP) systems [3] [5] provide basic theoretical models and meaningful tools for applications. In these models, the server plays the role of a *prover*, trying to convince the client who plays the role of a *verifier* that the result is correct. Based on probabilistic proof systems, many solutions have been proposed, which are efficient in terms of asymptotic complexity. Motivated by the delegation of computation [28], IP systems have been used to assure the client that an untrusted prover has actually performed the correct computation [28] [16] [17]. One notable variant of the PCP model is the idea of *argument systems* [14], which hold a more practical assumption that, in addition to the verifier being polynomial-time probabilistic, the prover is also computationally bounded. Recent breakthroughs in argument systems [32] [41] [42] [43] have made PCP-based approaches more practical.

The ultimate goal of all these methods is to ensure that the amount of verification workload performed by the client is less than the workload of performing the same computation from scratch. Although recent solutions show encouraging results, making verification closer to practicality than ever, the workload of verification remains quite expensive, especially for those cases requiring large-scale verification

(such as when large amounts of computation need to be outsourced, and then the results verified). But the average users may not be willing to spend their valuable time and resources on verification work, even though this new computational task is much less demanding than the original one, and neglect verification altogether. We can hardly imagine a hand-held device user devote CPU time and wireless bandwidth to verification.

In the spirit of outsourcing computation, a natural idea is to also outsource the verification. For this purpose, the client may delegate the verification to a third party – *the verifier*. The verifier does not need to be as powerful as the server doing the original computation. In the pay-per-use paradigm, the client should pay the verifier far less than the prover.

In addition to this novel verification-outsourcing paradigm, third-party verification may benefit other, equally-important applications. For example, disputes between the server and the client can be solved by an arbitrator who plays the role of the third-party verifier. Similar verifications may be required by government agencies, nonprofit organizations, and consumer organization, for the purpose of quality evaluation, project management, etc.

However, outsourcing verification is not trivial to implement. Several challenges emerge when outsourcing verification to untrusted verifiers. One of these, and the main focus of this paper, is the confidentiality concern. The results of computing are often confidential. Moreover, in many instances, even the details of the computation task itself may constitute sensitive material.

To the best of our knowledge, there is no feasible solution to these challenges. The two-party verification schemes cannot be directly adopted, either. Recomputing requires the verifier to have the same resources as the prover. If the prover provides a traditional NP-proof, the verifier is able to verify the result with fewer resources than required to recompute. But he still needs to read the entire proof, which costs polynomial time in the size of the computational task. Apart from the high cost, recomputing or checking NP-proofs provides little defense against curious verifiers.

In IP-based or PCP-based two-party verification schemes, it is necessary for the verifier to have perfect knowledge of the computation task and the result (in the context of computational complexity, the verifier needs to know the instance of the problem). If the third party simply runs the verifier's algorithms according to these two-party designs, the computation task and the result cannot be protected unless an expensive fully-homomorphic encryption system (e.g. [26]) is deployed.

The challenge here is how a third party can verify the correctness of the result *without* knowing the computation task and the result and without using expensive fully-homomorphic encryption [26]. In this paper, we describe a secure third-party confidentiality-preserving verification scheme.

Our work is related to, but different from delegation of computation to two or more servers [15] [7] [23] [16], where multiple servers with the same computational power compute individually and compete to convince the client to accept their results. In our design, without performing the same computation as the prover, the third party only needs fewer resources to verify the result from the prover.

The rest of this paper is structured as follows. We formally present the problem in Section 2. We start our demonstration with a brief exposition of current argument systems as the preliminaries in Section 3, since our protocols are related to the works of [32] [42] [43]. In Section 4, we propose our basic scheme, which enables the third party to verify the computational result without knowing the computational task. Then, in Section 5, we apply the additive homomorphic encryption technique to the basic scheme and describe the full solution to outsource the verification to an honest-but-curious verifier while protecting the secrecy of both the computation task and the input/output of corresponding circuit. In Section 6, we demonstrate the practical usage, and analyze the complexity of our design. Analysis shows the efficiency of our design. Section 7 concludes our findings and directs our future work.

2. PROBLEM STATEMENT

2.1 System Model

In the context of cloud computing, we propose a computation architecture involving three different parties: the client $\mathcal{C}$, who is computationally weak, has computation tasks to be delegated to the cloud; the cloud server $\mathcal{P}$, who is computationally powerful, provides computing services to the client; the verifier $\mathcal{V}$, who is not required to be computationally powerful, provides verification services, helping $\mathcal{C}$ to check the results computed by $\mathcal{P}$. $\mathcal{P}$ also plays the role of the prover that attempts to convince $\mathcal{C}$ (through $\mathcal{V}$) that the result is correct.

The computation tasks are formalized into the *arithmetic circuit satisfiability problem* – i.e., the *Circuit-SAT* problem over an arithmetic circuit. This problem is NP-complete, hence any other NP problem can be deterministically and efficiently reduced to it. The reason we choose this arithmetic circuit version instead of the original Boolean Circuit-SAT is that most real-world computation tasks can be easily mapped to arithmetic circuits. Let x be a single-output, n-gate arithmetic circuit (n includes the input gates and the output gate). In the language of computational complexity, we can consider x as an instance of the *arithmetic circuit satisfiability problem*. Formally, $x \in L$ where L is the language of the arithmetic circuit satisfiability problem. If x is satisfiable for a given output value $E \in \mathbb{F}$, then there exists an input y of length $|y| = m$ to the circuit x, which results in the circuit outputting E. This translates into a *correct assignment* z of the set of the outputs of all the gates in x. The assignment z can be in fact the concatenation of the input y with all the intermediate results inside the circuit, and has length $|z| = n$.

$\mathcal{C}$ is providing $\mathcal{P}$ with an output $E \in \mathbb{F}$, and expects $\mathcal{P}$ to return the input y which makes the circuit output E.

2.2 Threat Model

The threats faced by a client in our outsourced verification scenario come from malicious behaviors of both the prover $\mathcal{P}$ and the verifier $\mathcal{V}$. We assume $\mathcal{P}$ and $\mathcal{V}$ do not collude. This assumption is commonly used in multi-prover scenarios [15] [7] [23] [16]. Similar to previous proof systems, $\mathcal{P}$ can provide wrong responses to any queries, trying to cheat $\mathcal{C}$. In this paper, we only address the problem caused by an "honest-but-curious" $\mathcal{V}$ – one that is interested in learning the computation task and/or the result, but performs the protocol faithfully. Different attack models such as dishonest $\mathcal{V}$ will be addressed in future work.

We need to point out that in all of our schemes we omit the authentication part, since authentication is a mature technology and it is not within the scope of our paper. In our context, we assume that all parties are appropriately authenticated.

2.3 Design Goals

First of all, our proposed scheme should provide defense against the curious verifier $\mathcal{V}$ under the aforementioned model. To enable correct and efficient outsourcing of verification, the proposed scheme should satisfy the following requirements:

- *Correctness:* If y is the correct result, $\mathcal{P}$ can always construct a proof and convince the client $\mathcal{C}$ and the verifier $\mathcal{V}$ of the correctness of y.
- *Soundness:* If y is not the correct result, then for any proofs provided by a malicious $\mathcal{P}^*$, the probability that $\mathcal{V}$ wrongly accepts is negligibly small.
- *Efficiency:* The overall workload for the client $\mathcal{C}$ should be less – in an *amortized* sense (we will detail it later in Section 6) – than performing the verification himself. The workload for the verifier $\mathcal{V}$ should be comparable to that for verification in current two-parties designs [32, 42, 43]. Naturally, the workload for $\mathcal{V}$ should be far less than recomputing the result from scratch.

3. PRELIMINARIES

3.1 Probabilistically Checkable Proofs (PCPs)

In the PCP model, the verifier, a probabilistic polynomial-time (PPT) algorithm $\mathcal{V}$ can be convinced by a prover $\mathcal{P}$ that a string x belongs to a language L in an interactive way: $\mathcal{V}$ has random access to the proof π which is constructed by $\mathcal{P}$. By querying π (accessing the proof and reading several values), $\mathcal{V}$ will either accept or reject. *Correctness:* If $x \in L$, $\mathcal{P}$ can always construct a proof π such that $\mathcal{V}$ will accept that $x \in L$. We call π the *correct proof* for x. *Soundness:* If $x \notin L$ then for any π, the probability that $\mathcal{V}$ wrongly accepts is less than a constant ϵ. Let L be any language. The PCP theorem [6] [5] [3] [2] [18] guarantees that, if $L \in NP$, then with only a constant number of queries, $\mathcal{V}$ can verify $x \in L$ with negligible error probability (soundness).

Early results of PCP [2] [3] [5] [7] [12] [12] [34] [35] [37] were viewed as important discoveries only in the theory of computational complexity. Recent research focuses on the efficiency [9] [36], length [9] [8] [10] [18] [39] or soundness error [19] [38] [20] [21] of PCPs.

3.2 Homomorphic Encryption

The commitment protocols of existing efficient argument systems are all based on homomorphic encryption. This homomorphic encryption does not refer to fully-homomorphic encryption [26]. Only additive homomorphism is used in current argument systems: that is, the ciphertext of the result of adding two plaintexts can be efficiently computed from the ciphertexts of the two plaintexts. Formally, for any valid ciphertexts $c_1 = \texttt{Enc}(pk, m_1)$ and $c_2 = \texttt{Enc}(pk, m_2)$, there is an efficient algorithm $\mathcal{H}$ such that $\mathcal{H}(c_1, c_2) = \texttt{Enc}(pk, m_1 + m_2)$, where pk is the public key and m_1, m_2 are plaintexts. In this paper, the underlying homomorphic encryption is assumed to be semantically secure [29].

3.3 Efficient Arguments without Short PCPs

Arguments [14] are interactive proof systems, consisting of two PPT algorithms: the prover $\mathcal{P}$ and the verifier $\mathcal{V}$. For an NP language L with soundness error $\epsilon(\cdot)$, an argument is both *complete* and *sound* if it satisfies the following conditions: (a) Completeness: for any $x \in L$ and corresponding NP witness w, the interaction between $\mathcal{V}(x)$ and $\mathcal{P}(x, w)$ leads $\mathcal{V}$ to accept the proof as true. (b) Soundness: for any $x \notin L$, and any efficient prover $\mathcal{P}^*$, the interaction between $\mathcal{V}(x)$ and $\mathcal{P}^*(x)$ leads $\mathcal{V}$ to accept the proof with probability less than $\epsilon(|x|)$.

To make argument systems efficient, current implementations rely on PCPs. However, PCP algorithms assume that the proof is computed by the prover, and fixed before the interaction with the verifier begins. The same assumption cannot be made in the context of argument systems. To bridge the gap between arguments and PCPs, an additional protocol is required, in which the prover commits to the proof before starting the PCP protocol with the verifier. Consequently, an argument is generally formed by joining together two protocols: a *PCP* and a *commitment*. Since the commitment protocol should maintain the efficiency of the argument, it is generally not feasible to require $\mathcal{P}$ to send the entire PCP proof to $\mathcal{V}$ due to the length of the proof. Two solutions can be implemented to overcome this obstacle: (1) make the PCP proof short, and (2) use cryptographic techniques to enable even shorter commitments to these short proofs. One of the first efforts in the latter direction is that of [37], which proposed to use a Merkle hash-tree construction to enable the prover to efficiently commit to the proof. Implicitly, the security of the protocol is bound to the security of the underlying hash function. To avoid the need for convoluted short PCP proofs, as well as the uncertain security of practical hashing primitives, [32] takes a new approach to argument systems: maintain a large (exponential-size) proof, and base the commitment on (computationally) provably-secure encryption primitives – public-key primitives.

The protocols of [32] are restricted to linear PCPs ([2], Section6). It is shown how SAT problems, formulated in the context of a boolean circuit, can be readily addressed by a simple linear PCP [32]. To form the argument system, [32] complemented the linear PCP with the notion of *commitment with linear decommitment*, which is instantiated with a simple public-key-based protocol. Since our work is closely related to that of [32], we will provide both the definition of *commitment with linear decommitment*, and a brief sketch of its instantiation in this section.

DEFINITION 1. *Commitment with Linear Decommitment ([32])* A commitment with linear decommitment *(in the context of argument systems) is a protocol between the prover $\mathcal{P}$ and verifier $\mathcal{V}$ – both modeled as interactive PPT algorithms – consisting of a commitment phase, and a decommitment phase, and aiming to securely commit the prover to a linear function $f_d : \mathbb{F}^n \to \mathbb{F}$ expressed as $f_d(z) = \langle d, z \rangle$, where $d, z \in \mathbb{F}^n$, and $\langle d, z \rangle$ is the natural inner (dot) product over vector spaces. In the commitment phase, an environment $\mathcal{E}$ gives $\mathcal{P}$ inputs d and $\mathbb{F}$, and gives $\mathcal{V}$ inputs $\mathbb{F}$ and the arity n. The interaction between $\mathcal{P}$ and $\mathcal{V}$ results in decommitment information z_P and z_V, respectively. In the decommitment phase, $\mathcal{E}$ gives $\mathcal{P}$ a decommitment query $q \in \mathbb{F}^n$. After further interaction between $\mathcal{P}$ and $\mathcal{V}$, the verifier $\mathcal{V}$ outputs either a value $a \in \mathbb{F}$, or the symbol $\perp$ (reject).*

A commitment with linear decommitment has the following properties. (a) Correctness: for any n and $\mathcal{E}$ generating d, q, at the end of the decommitment phase, the verifier outputs $a = f_d(q)$. (b) Binding: for the same decommitment information z_P, z_V (obtained after the commitment phase) and environment inputs q in the decommitment phase, the probability that at the end of the protocol the verifier outputs two different values (a_1, a_2) is negligible in n.

Ishai et al. [32] took L as the satisfiability problem over an arithmetic circuit to show how to construct a correct proof for any arithmetic circuit and how to verify this circuit is satisfiable. Since this problem is NP-complete, every other NP problems can be deterministically and efficiently reduced to it. The PCP theorem guarantees that, if $L \in NP$ then with only constant number of queries, $\mathcal{V}$ can verify $x \in L$ with negligible error probability (soundness).

The instance x is an arithmetic circuit in this problem. For $x \in L$, there is a correct assignment z of the inputs to all gates in x. z can be also viewed as values of both the input of x and intermediate results. The correct proof is an exponential size PCP, which consists of two substrings. Each of the substrings can be viewed as a linear function: $\pi^{(1)} : \mathbb{F}^n \mapsto \mathbb{F}$ and $\pi^{(2)} : \mathbb{F}^{n^2} \mapsto \mathbb{F}$ where n is the length of a correct assignment z, $\pi^{(1)}(\cdot) = \langle z, \cdot \rangle$ and $\pi^{(2)}(\cdot) = \langle z \otimes z, \cdot \rangle$. Here, $\langle u, v \rangle$ denotes the inner product of two vectors u and v, and $u \otimes v$ denotes the outer product of two vectors u and v. The outer product is equivalent to a matrix multiplication uv^T, provided that u and v are both represented as a column vector. The whole proof string can be viewed as one single linear function $\pi : \mathbb{F}^{n^2+n} \mapsto \mathbb{F}$ such that $\pi(\cdot) = \langle z || z \otimes z, \cdot \rangle$ where $z||z \otimes z$ is the concatenation of the two vectors z and $z \otimes z$. When $\mathcal{V}$ sends the query q to π, he will get back $\pi(q)$. For $q \in \mathbb{F}^n$, $\pi(q) = \langle z||z \otimes z, q||0^{n^2} \rangle = \langle z, q \rangle$. For $q \in \mathbb{F}^{n^2}$, $\pi(q) = \langle z||z \otimes z, 0^n||q \rangle = \langle z \otimes z, q \rangle$. For $q \in \mathbb{F}^{n^2+n}$, $\pi(q) = \langle z||z \otimes z, q \rangle$.

As in the first column of Table 1, the commitment protocol was designed in [32], where a commitment to a proof is constructed and $\mathcal{V}$ can verify that the proof he queries is committed to a linear function.

Once the proof is committed, $\mathcal{V}$ will check the proof in the linear PCP fashion. The verification consists of three kinds of tests. The first is the linearity test. $\mathcal{V}$ picks at random $q_1, q_2 \in \mathbb{F}^n$ and verifies $\pi(q_1) + \pi(q_2) = \pi(q_1 + q_2)$. The second is the quadratic consistency test. $\mathcal{V}$ picks at random $q_3, q_4 \in \mathbb{F}^n$ and verifies $\pi(q_3) \cdot \pi(q_4) = \pi(q_3 \otimes q_4)$. The third is the circuit correctness test. Each gate implies a constraint. For each constraint f_u, $(u = 1, 2, \cdots n)$, $\mathcal{V}$ picks at random a weight δ_u and constructs the weighted sum $\sum_{u=1}^{n} \delta_u f_u$. The sum can be rewritten as $\pi(q_5) + c = 0$, $c \in \mathbb{F}$. If each constraint is satisfied, the weighted sum of the constraints $\pi(q_5) + c = 0$ is also satisfied. If there are some constrains not satisfied, the probability that the $\pi(q_5) + c = 0$ is $1/|\mathbb{F}|$. All these tests can be performed several times to drive the error probability down.

3.4 Two Recent Efficient Arguments: PEPPER and GINGER

Several recent works build upon the ideas developed in [32]. Of these, [42] and [43] are the most relevant to our work. To bring the protocol of [32] closer to practicality, [42] introduces a new protocol called PEPPER. It first shows that large savings in both computation and communication overhead can be achieved by expressing the SAT problem in the format of arithmetic circuits with *concise gates* [42] instead of the boolean circuits of [32]. In addition, by batching together multiple queries (to the same committed function), [42] can decommit all of them in a single commit-decommit round, rather than providing separate decommitments for each query.

In Ishai et al.'s original commitment design [32], one query is accompanied by an auxiliary query which is associated to a commitment. This requires many commitments, therefore increases the overhead. In [42], one auxiliary query is made, which is a random linear combination of all the PCP queries and the secret information that is associated to the commitment. In this design, one decommitment can guarantee many PCP queries are bound to the committed function. This sharply reduced the computational cost of generating the commitment information (although remaining cost is still very high.). The Single-Commit-Multi-Decommit design is demonstrated in the second column of Table 1.

Finally, by batching together multiple computations, [42] only requires a single random commitment query r for all computations involved (rather than a different r for each computation), hence achieving great savings in the encryption process – recall that the query r is transmitted to the prover after it has been encrypted by the homomorphic encryption algorithm.

Building on top of [42], additional improvements are provided in [43], in the context of a more efficient protocol called GINGER. A thorough analysis of PEPPER pointed out that the linearity tests are superfluous [43], and that in fact the commitment protocol alone guarantee the linearity of the proof. In addition, several queries of the quadratic correction test may be omitted [43] from the $\mathcal{V}$-to-$\mathcal{P}$ transmission, as they can be easily computed by the prover from the remaining quadratic correction queries.

Definition 2. A commitment to a function with multiple decommitments (CFMD)(from [42]) *A commitment to a function with multiple decommitments (CFMD) is defined by a pair of PPT algorithms $(\mathcal{P}, \mathcal{V})$ (a sender and receiver, which correspond to our prover and verifier) anticipating the following experiment with an environment $\mathcal{E}$. $\mathcal{E}$ generates $\mathbb{F}, w$ and $Q = (q_1, \cdots, q_\mu)$. The two phases are:*

- Commitment phase: *$\mathcal{P}$ has w, and $\mathcal{P}$ and $\mathcal{V}$ interact, based on their random inputs.*
- Decommitment phase: *$\mathcal{E}$ gives Q to $\mathcal{V}$, and $\mathcal{P}$ and $\mathcal{V}$ interact again, based on further random inputs. At the end, $\mathcal{V}$ outputs $A = (a_1, \cdots, a_\mu) \in \mathbb{F}^\mu$ or $\perp$.*

A commitment to a function with multiple decommitments (CFMD) *should satisfy the following properties:*

- ***Correctness:*** *at the end of the decommitment phase, $\mathcal{V}$ outputs $\pi(q_i) = \langle w, q_i \rangle$, (for all i), if $\mathcal{P}$ is honest.*
- ***ϵ_B-Binding:***. *Consider the following experiment. The environment $\mathcal{E}$ produces two (possibly distinct) μ-tuples of queries: $Q = (q_1, \cdots, q_\mu)$ and $\hat{Q} = (\hat{q_1}, \cdots, \hat{q_\mu})$. $\mathcal{V}$ and a cheating $\mathcal{P}^*$ run the commitment phase once and two independent instances of the decommitment phase. In the two instances $\mathcal{V}$ presents the queries as Q and $\hat{Q}$, respectively. We say that $\mathcal{P}^*$ wins if $\mathcal{V}$'s outputs at the end of the respective decommit phases are*

$A = (a_1, \cdots, a_\mu)$ and $\hat{A} = (\hat{a_1}, \cdots, \hat{a_\mu})$, and for some i, j, we have $q_i = \hat{q}_j$ but $a_i \neq \hat{a_j}$. The protocol holds the ϵ_B-Binding property if for all $\mathcal{E}$ and for all efficient $\mathcal{P}^*$, the probability of $\mathcal{P}^*$ winning is at most ϵ_B. The probability is taken over three sets of independent randomness: the commitment phase and the two runnings of the decommitment phase.

4. BASIC SCHEME: VERIFICATION WITHOUT CIRCUIT INFORMATION

We present a basic version of our protocol in this section. We then bootstrap the process to develop the full solution in next section. In this basic scheme, the client $\mathcal{C}$ delegates the verification task to the verifier $\mathcal{V}$ and $\mathcal{V}$ can verify the proof without knowing the computational task, i.e., the underlying arithmetic circuit x. Our basic scheme is designed by joining together two protocols: a novel linear PCP and a new commitment protocol. Recall that PCP systems assume the proof is computed by $\mathcal{P}$, and fixed before the interaction with the $\mathcal{V}$ begins. The same assumption cannot be made in Cloud Computing. For efficiency reasons, it is also not feasible to require $\mathcal{P}$ to send the entire PCP proof to $\mathcal{V}$. It is the commitment protocol that guarantees $\mathcal{P}$ commits to the proof before starting the PCP protocol with $\mathcal{V}$.

4.1 A Building Block: A New Commitment Protocol

In the context of Circuit-SAT problem over an arithmetic circuit, we propose the following new commitment protocol for linear PCP system. It is a two-party protocol between the prover $\mathcal{P}$ and another party denoted by $\mathcal{C}/\mathcal{V}$ (as in "client/verifier"). We do not differentiate between the client $\mathcal{C}$ and the verifier $\mathcal{V}$ in this subsection. This separation will be done in next subsection. Recall that the Circuit-SAT problem is to find an input $y = (y_1, y_2, \cdots, y_m)$ which makes the circuit x output a given value E. The arithmetic circuit x consists of $n = |x|$ arithmetic gates. Each gate implies a constraint $f_u, 1 \leq u \leq n$ as follows:

- $f_u(z_u) = z_u$ for $1 \leq u \leq m$. These are the constraints for the input gates.
- $f_u(z_i, z_j, z_k) = 0$ for $m+1 \leq u \leq n-1$, where f_u is a linear or quadratic polynomial of z_i, z_j, z_k. Here z_i, z_j, z_k are the two inputs and one output of a certain gate of x.
- $f_n(z_n) = E$. This is the constraint for the output gate.

Our commitment protocol rearranges the argument system to put the circuit-dependent portions inside the commitment phase (the offline stage). This approach not only simplifies the client/verifier's operation on the verification side, but also provides circuit-secrecy against the verifiers while outsourcing the verification tasks.

Our commitment protocol is demonstrated in the third column of Table 1. This protocol eventually includes two decommitment processes, one is from Step 4 to Step 7, the other is from Step 8 to Step 11.

We will prove that after the commitment construction phase, all of $\mathcal{P}$'s answers to later queries that pass both the decommitment checks are guaranteed to be bound to one single function (from queries to answers) with high probability. That is, having committed, $\mathcal{P}$ is very likely incapable of cheating the verifiers with fake answers. Moreover, this function is guaranteed to be linear with high probability.

THEOREM 1. (*Main Theorem) For our commitment protocol, the following holds. For any environment $\mathcal{E}$, for any query q in either of the decommitment phases, the corresponding answer accepted by $\mathcal{V}$ at the end of the protocol is guaranteed to be the function value $\tilde{\pi}(q)$ except with probability less than $\frac{1}{|\mathbb{F}|} + neg(n)$, where $\tilde{\pi}(q)$ is a linear function, $neg(n)$ is a negligible function, and the probability is over all randomness of $\mathcal{P}^*$ and $\mathcal{V}$ in all phases.*

We prove this theorem in Appendix of the full version [1].

4.2 A Delegation-of-Verification Scheme with Partial Circuit Confidentiality

As in the context of cloud computing, $\mathcal{C}$ sends the circuit description x to $\mathcal{P}$. After finding out the solution y with his powerful computation ability, $\mathcal{P}$ returns y to $\mathcal{C}$. Before outsourcing the verification task, $\mathcal{C}$ constructs the commitment according to our basic commitment scheme. $\mathcal{C}$ plays the role of $\mathcal{C}/\mathcal{V}$ in that commitment construction protocol and gets: w_0, r_1, R_1, r_0, R_0, s_1, S_1, s_0 S_0. The decommitment phases are a little bit different from our basic commitment scheme. $\mathcal{C}$ generates a random value $\alpha_0 \in \mathbb{F}$ and computes $(r_1||R_1) + \alpha_0(r_0||R_0)$. Then, $\mathcal{C}$ outsources the verification task to a third party $\mathcal{V}$. $\mathcal{C}$ sends w_0, $(r_1||R_1) + \alpha_0(r_0||R_0)$, s_1, S_1, $s_1 + S_1 + \alpha_0(s_0 + S_0)$, E, and y_i's $(i = 1, \cdots m)$ to $\mathcal{V}$. Later, $\mathcal{V}$ will perform the decommitment.

Since our commitment protocol has inherently provided the linearity test, (see the full version for details [1]) it is sufficient to conduct only circuit and quadratic consistency tests. For the circuit test, $\mathcal{V}$ generates w_1 and w_2 as in Step 4 and Step 5 of our basic commitment scheme. As in Step 5 and Step 6, $\mathcal{V}$ queries $\mathcal{P}$ with w_1 and w_2, receives back A_1, a_1, B_1 and b_1. Then $\mathcal{V}$ checks the equations as in Step 7. $\mathcal{V}$ will also check the circuit correctness, i.e., whether

$$A_1 + a_1 = \sum_{u=1}^{m} y_u w_{1u} + w_{1n} E. \tag{4.1}$$

For the quadratic consistency tests, $\mathcal{V}$ first conducts the second decommitment according to Steps 8, 9, 10, 11. Here, $\mathcal{V}$ uses only three testing queries (that is, $\mu = 3$) and the decommit query t. $\mathcal{V}$ randomly generates queries q_2, q_3 both from $\mathbb{F}^n$. He randomly generates $\alpha_2, \alpha_3, \alpha_4$, all from $\mathbb{F}$. He constructs the following queries: $q_4 = q_2 \otimes q_3$, and $t = (r_1||R_1 + \alpha_0(r_0||R_0)) + \sum_{i=2}^{3}(q_i||0^{n^2}) + \alpha_4(0^n||q_4)$, where 0^u is the u-dimension zero vector. $\mathcal{V}$ queries $\mathcal{P}$ with (q_2, q_3, q_4, t). $\mathcal{P}$ returns (a_2, a_3, a_4, b_2) where $a_2 = \langle q_2, z\rangle$, $a_3 = \langle q_3, z\rangle$, $a_4 = \langle q_4, z \otimes z\rangle$, $b_2 = \langle t, z \otimes z\rangle$. At the second decommitment, $\mathcal{V}$ checks whether $b_2 = (s_1 + S_1 + \alpha_0(s_0 + S_0)) + \sum_{i=2}^{4} \alpha_i a_i$. For quadratic consistency, $\mathcal{V}$ checks whether $a_4 = a_2 a_3$.

If all the checks pass, $\mathcal{V}$ will instruct $\mathcal{C}$ to accept. Otherwise, $\mathcal{V}$ instructs $\mathcal{C}$ to reject.

4.3 Theoretical Analysis: Correctness and Soundness

It is easy to see that without knowing the circuit x, $\mathcal{V}$ conducts all the PCP checks (except the linearity tests, since the commitment has provided linearity tests already) for $\mathcal{C}$. The correctness and soundness of this scheme follows directly from the linear PCP scheme. However, it should be noted that $\mathcal{V}$ has access to the pair $((r_1||R_1) + \alpha_0(r_0||R_0), s_1 +$

Table 1: Comparison of Commitment Protocols

Ishai et al. [32]	GINGER [43]	Our Basic Commitment Scheme
Commitment Phase	**Commitment Phase**	**Commitment Phase**
Prover's Input: a vector $d \in \mathbb{F}^{n^2+n}$, a linear function $\pi : \mathbb{F}^{n^2+n} \to \mathbb{F}$ where $\pi(q) = \langle q, d \rangle$.	**Prover's Input:** $z \in \mathbb{F}^n$, a linear function $\pi : \mathbb{F}^{n^2+n} \mapsto \mathbb{F}$ where $\pi(\cdot) = \langle z \| \| z \otimes z, \cdot \rangle$.	**Prover's Input:** a vector $z \in \mathbb{F}^n$, a linear function $\pi : \mathbb{F}^{n^2+n} \mapsto \mathbb{F}$ where $\pi(\cdot) = \langle z \| \| z \otimes z, \cdot \rangle$, n is the length of a correct assignment z.
Verifier's Input: arity n^2+n, security parameter k for the homomorphic encryption.	**Verifier's Input:** arity n, security parameter k of the encryption.	**Verifier's Input:** arity n, security parameter k of the encryption, the circuit x, the circuit's input $y = (y_1, \cdots, y_m)$ and output E.
Step 1: $\mathcal{V}$ generates the key pair $(pk, sk) \leftarrow \texttt{Gen}(1^k)$ and $r = (r_1, \cdots, r_{n^2+n}) \in_R \mathbb{F}^{n^2+n}$, $r_i \in \mathbb{F}$, $i = 1, \cdots, n^2+n$. $\mathcal{V}$ encrypts each entry of r and sends $\texttt{Enc}(pk, r_1), \cdots, \texttt{Enc}(pk, r_{n^2+n})$ to $\mathcal{P}$.	**Step 1:** $\mathcal{V}$ generates the key pair $(pk, sk) \leftarrow \texttt{Gen}(1^k)$ and $r = (r_1, \cdots, r_{n^2+n}) \in_R \mathbb{F}^{n^2+n}$. $r_i \in \mathbb{F}$, $i = 1, \cdots, n^2+n$. $\mathcal{V}$ encrypts each entry of r and sends $\texttt{Enc}(pk, r_1), \cdots, \texttt{Enc}(pk, r_{n^2+n})$ to $\mathcal{P}$.	**Step 1:** $\mathcal{C}/\mathcal{V}$ randomly picks $r_0 \in \mathbb{F}^n$ and $R_0 \in \mathbb{F}^{n^2}$ and constructs the corresponding commitments (s_0, S_0) according to Ishai et al.'s commitment protocol [32].
Step 2: $\mathcal{P}$ makes use of the homomorphism of $\texttt{Enc}$ and gets $e = \texttt{Enc}(pk, \langle r, d \rangle)$. $\mathcal{P}$ sends e to $\mathcal{V}$.	**Step 2:** Using the homomorphism, $\mathcal{P}$ gets: $e = \texttt{Enc}(pk, \langle r, z \rangle)$ and sends it to $\mathcal{V}$.	**Step 2:** $\mathcal{C}/\mathcal{V}$ randomly generates an n-dimension weight vector $w_0 = (w_{01}, w_{02}, \ldots, w_{0n}) \in \mathbb{F}^n$, where each entry corresponds to a constraint f_u $(u = 1, \cdots, n)$ of the arithmetic circuit x. $\mathcal{C}/\mathcal{V}$ multiplies each constraint f_u of the circuit by w_{0u}, $u = 1, \cdots, n$ and constructs their summation as $\sum_{u=1}^{n} w_{0u} f_u = \sum_{u=1}^{m} w_{0u} y_u + w_{0n} E$. The summation of all these weighted constraints can be rewritten as $\langle R_1, z \otimes z \rangle + \langle r_1, z \rangle = c_0$, where $c_0 = \sum_{u=1}^{m} y_u w_{0u} + w_{0n} E$.
Step 3: $\mathcal{V}$ decrypts e and gets $s = \langle r, d \rangle = \texttt{Dec}(sk, e)$. (s, r) will be kept for decommitment.	**Step 3:** $\mathcal{V}$ receives e. He gets $s = \langle r, z \rangle = \texttt{Dec}(sk, e)$. (s, r) will be kept for decommitment.	**Step 3:** Using R_1 and r_1, $\mathcal{C}/\mathcal{V}$ constructs the corresponding commitments (s_1, S_1) according to Ishai's commitment protocol [32].
Decommitment Phase	**Decommitment Phase**	**Decommitment Phase**
Prover's Input: d, π	**Prover's Input:** z, π, n	**Prover's Input:** x, z including y and E, π, n.
Verifier's Input: arity n^2+n, a PCP query q, decommitment information (r, s).	**Verifier's Input:** arity n, μ PCP queries $q_1, \cdots, q_\mu$, decommitment information (r, s).	**Verifier's Input:** n, μ PCP queries $q_1, \cdots, q_\mu$, decommitment information $(w_0, r_0, R_0, r_1, R_1, s_0, S_0, s_1, S_1)$.
Step 4: $\mathcal{V}$ picks at random a secret $\alpha \in_R \mathbb{F}$.	**Step 4:** $\mathcal{V}$ picks μ secrets $\alpha_1, \cdots, \alpha_\mu \in \mathbb{F}$	**Step 4:** $\mathcal{C}/\mathcal{V}$ generates randomly an n-dimension weight vector $w_1 = (w_{11}, \ldots, w_{1n}) \in \mathbb{F}^n$ and a secret $\alpha_1 \in \mathbb{F}$.
Step 5: $\mathcal{V}$ sends $q, r + \alpha q$ to the prover.	**Step 5:** $\mathcal{V}$ queries $\mathcal{P}$ with $q_1, \cdots, q_\mu$ and $t = r + \sum_{i=1}^{\mu} \alpha_i q_i$.	**Step 5:** $\mathcal{C}/\mathcal{V}$ queries $\mathcal{P}$ with vector w_1 and $w_2 = w_0 + \alpha_1 w_1$.
Step 6: $\mathcal{P}$ responds with 2 values that are in $\mathbb{F}$: (a, b) where a is supposed to be $\pi(q)$ and b is supposed to be $\pi(r + \alpha q)$.	**Step 6:** $\mathcal{P}$ returns $\mu + 1$ values: $(a_1, \cdots, a_\mu, b)$ where $a_i = \pi(q_i)$ for $i = 1, \cdots, \mu$ and $b = \pi(t)$	**Step 6:** From w_1, $\mathcal{P}$ constructs the weighted summation of all constraints just like what $\mathcal{C}/\mathcal{V}$ does in Step 2 and gets $\langle Q_0, z \otimes z \rangle + \langle q_0, z \rangle = \sum_{u=1}^{m} y_u w_{1u} + w_{1n} E$. From it, $\mathcal{P}$ learns Q_0 and q_0 and returns: $A_1 = \langle Q_0, z \otimes z \rangle$ and $a_1 = \langle q_0, z \rangle$. Similarly, from w_2, $\mathcal{P}$ constructs the weighted summation $\langle T_1, z \otimes z \rangle + \langle t_1, z \rangle = \sum_{u=1}^{m} y_u w_{2u} + w_{2n} E$ and learns T_1 and t_1. $\mathcal{P}$ returns: $B_1 = \langle T_1, z \otimes z \rangle$ and $b_1 = \langle t_1, z \rangle$.
Step 7: $\mathcal{V}$ will determine whether $b = s + \alpha a$. If it holds, the $\mathcal{V}$ will accept and output a; otherwise it will reject and output $\perp$.	**Step 7:** $\mathcal{V}$ checks whether $b = s + \alpha_1 a_1 + \cdots \alpha_\mu a_\mu$ holds. If so, $\mathcal{V}$ outputs $a_1, \cdots, a_\mu$. Otherwise, he rejects and output $\perp$.	**Step 7:** $\mathcal{C}/\mathcal{V}$ checks whether $b_1 = s_1 + \alpha_1 a_1$ and $B_1 = S_1 + \alpha_1 A_1$. If both hold, $\mathcal{C}/\mathcal{V}$ goes on to Step 8. Otherwise, $\mathcal{C}/\mathcal{V}$ rejects the proof.
		Step 8: $\mathcal{C}/\mathcal{V}$ randomly generates $\alpha_0, \alpha_1, \cdots, \alpha_\mu$ from $\mathbb{F}$. $\mathcal{C}/\mathcal{V}$ constructs $t = (r_1 \| \| R_1) + \alpha_0 (r_0 \| \| R_0) + \sum_{k=1}^{\mu} \alpha_k q_k$.
		Step 9: $\mathcal{C}/\mathcal{V}$ queries $\mathcal{P}$ with $q_1, \cdots, q_\mu$ and t.
		Step 10: $\mathcal{P}$ returns $\mu + 1$ corresponding answers $(a_1, \cdots, a_\mu, b_2)$ where for $k = 1, 2, \cdots \mu$, $a_k = \langle q_k, z \| \| z \otimes z \rangle$ and $b_2 = \langle t, z \| \| z \otimes z \rangle$.
		Step 11: $\mathcal{C}/\mathcal{V}$ checks whether $b_2 = (s_1 + S_1 + \alpha_0 (s_0 + S_0)) + \sum_{k=1}^{\mu} \alpha_k a_k$ holds. If so, $\mathcal{C}/\mathcal{V}$ accepts. Otherwise $\mathcal{C}/\mathcal{V}$ rejects.

$S_1 + \alpha_0(s_0 + S_0))$, which leaks information about the circuit. Hence the *partial circuit confidentiality* is afforded by this scheme. Nevertheless, building upon this scheme, full circuit confidentiality is achieved by the full solution outlined in the next section. Our basic scheme is also a distinct improvement over existing argument systems. During the verification procedure, the verifier does not need to read the circuit. He can generate all the queries with the cost of merely generating random numbers. By comparison, to generate a query, current argument systems need to both generate random numbers, and to calculate weighted summations of all circuit's constraints.

5. THE FULL SOLUTION TO DELEGATING VERIFICATION TO A CURIOUS VERIFIER

In certain scenarios, both the computation circuit and input/output of this circuit are sensitive. A curious verifier may be interested in information regarding the computation task (the circuit) and/or the circuit's input and output. In this section, we use the previously described basic scheme to develop the full solution against the curious verifier. The full version of the protocol consists of four phases, detailed in the following subsections: outsourcing computation, constructing commitments, outsourcing verification, and making a decision.

5.1 Outsourcing Computation Phase

$\mathcal{C}$ possesses an additive homomorphic cryptosystem. He generates a key pair (SK, PK). $\mathcal{C}$ sends the arithmetic circuit description x along with the public key PK to $\mathcal{P}$. After finding out the solution y with his powerful computation ability, $\mathcal{P}$ returns y to $\mathcal{C}$. After this computation, $\mathcal{P}$ obtains a correct assignment z of all the input of each gate in x. z can be viewed as the values of both y (the input of x) and intermediate results. $\mathcal{P}$ possesses a corresponding linear function: $\pi : \mathbb{F}^{n^2+n} \mapsto \mathbb{F}$ (remember $n = |x| = |z|$) such that, $\pi(\cdot) = \langle z||z \otimes z, \cdot\rangle$.

5.2 Constructing Commitments

Before outsourcing the verification task to a third party, $\mathcal{C}$ constructs the commitment according to the protocol described in Table 1. At the end of the construction, $\mathcal{C}$ possesses: w_0, r_1, R_1, r_0, R_0, and s_1, S_1, s_0, S_0. $\mathcal{C}$ generates a random value $\alpha_0 \in \mathbb{F}$ and computes $(r_1||R_1) + \alpha_0(r_0||R_0)$. After constructing the commitment, $\mathcal{C}$ randomly picks $w_{11}, w_{12}, \cdots, w_{1m}$ and w_{1n}, all in $\mathbb{F}$. Let w_1' be $((w_{11}, w_{12}, \cdots, w_{1m})||0^{n-m-1}||w_{1n})$. With these numbers, $\mathcal{C}$ computes $c_0 = \sum_{u=1}^{m} w_{1u}y_u + w_{1n}E$, then randomly generates α_1 and sends α_1, s_1,S_1, c_0, $w_0+\alpha_1 w_1'$, $(r_1||R_1)+\alpha_0(r_0||R_0)$, $\texttt{Enc}(PK, (s_1 + S_1 + \alpha_0(s_0 + S_0)))$, and the public key PK to $\mathcal{V}$. Meanwhile, $\mathcal{C}$ sends $w_{11}, \cdots, w_{1m}$, $w_{1n}$and PK to $\mathcal{P}$.

5.3 Outsourcing Verification Phase

In this phase, $\mathcal{V}$ will verify the result without knowing the circuit and any of the assignments z (including y) in a PCP fashion. Given that the commit/decommit protocol has inherently provided the linearity test, it is sufficient to conduct only circuit tests and quadratic consistency tests.

The first step is the circuit satisfiability test. As in Section 4, $\mathcal{V}$ generates randomly two weight vectors. However, the vectors are a little bit different here: he generates $(w_{1(m+1)}, w_{1(m+2)}, \cdots, w_{1(n-1)})$, all from $\mathbb{F}$. Let w_1'' be $(0^m||(w_{1(m+1)}, w_{1(m+2)}, \cdots, w_{1(n-1)})||0)$. He generates w_2 as $w_2 = (w_0+\alpha_1 w_1')+\alpha_1 w_1''$, and queries $\mathcal{P}$ with w_1'' and w_2. This time, he will receive back $\texttt{Enc}(PK, A_1)$, $\texttt{Enc}(PK, a_1)$, $\texttt{Enc}(PK, B_1)$, $\texttt{Enc}(PK, b_1)$. Using PK, $\mathcal{V}$ computes $\texttt{Enc}(PK, (s_1 + \alpha_1 c_0))$ from s_1, α_1 and c_0. Using the additive homomorphism of underlying encryption, $\mathcal{V}$ can compute $\texttt{Enc}(PK, (b_1 - ((s_1 + \alpha_1 c_0) + \alpha_1 a_1)))$ from $\texttt{Enc}(PK, (s_1 + \alpha_1 c_0))$, $\texttt{Enc}(PK, a_1)$, and $\texttt{Enc}(PK, b_1)$. $\texttt{Enc}(PK, (b_1 - ((s_1 + \alpha_1 c_0) + \alpha_1 a_1)))$ is denoted by $\texttt{Enc}(PK, plain_1)$. Similarly, he gets $\texttt{Enc}(PK, B_1 - (S_1 + \alpha_1 A_1))$, denoted by $\texttt{Enc}(PK, plain_2)$. Using the additive homomorphism of underlying encryption, from $\texttt{Enc}(PK, A_1)$ and from $\texttt{Enc}(PK, a_1)$, $\mathcal{V}$ computes $\texttt{Enc}(PK, A_1 + a_1)$, denoted by $\texttt{Enc}(PK, plain_3)$.

The second step is the quadratic consistency test. $\mathcal{V}$ randomly generates $\alpha_2, \alpha_3, \alpha_4$ all from $\mathbb{F}$. He randomly generates queries q_2, q_3 and constructs following queries: $q_4 = q_2 \otimes q_3$, $t = r_1||R_1 + \alpha_0(r_0||R_0) + \sum_{i=2}^{3} \alpha_i(q_i||0^{n^2}) + \alpha_4(0^n||q_4)$. $\mathcal{V}$ queries $\mathcal{P}$ with (q_2, q_3, q_4, t) and gets back $\texttt{Enc}(PK, a_2)$, $\texttt{Enc}(PK, a_3)$, $\texttt{Enc}(PK, a_4)$, and $\texttt{Enc}(PK, b_2)$ where $a_2 = \langle q_2, z\rangle$, $a_3 = \langle q_3, z\rangle$, $a_4 = \langle q_4, z \otimes z\rangle$, $b_2 = \langle t, z||z \otimes z\rangle$. We denote $\texttt{Enc}(PK, (b_2 - ((s_1 + S_1 + \alpha_0(s_0 + S_0)) + \sum_{i=2}^{4} \alpha_i a_i)))$ by $\texttt{Enc}(PK, plain_4)$, which can be computed from $\texttt{Enc}(PK, (s_1 + S_1 + \alpha_0(s_0 + S_0)))$, $\texttt{Enc}(PK, a_i)$'s $(i = 2, 3, 4)$ and $\texttt{Enc}(PK, b_2)$ using the homomorphism of the underlying cryptosystem. Then, $\mathcal{V}$ randomly generates four random numbers from $\mathbb{F}$: $\theta_1, \cdots, \theta_4$, and constructs $\texttt{Enc}(PK, \sum_{i=1}^{4} plain_i \cdot \theta_i)$ from $\texttt{Enc}(PK, plain_i)$'s using the homomorphism. $\mathcal{V}$ sends $\texttt{Enc}(PK, \sum_{i=1}^{4} plain_i \cdot \theta_i)$ to $\mathcal{C}$ with $\texttt{Enc}(PK, a_4)$, $\texttt{Enc}(PK, a_3)$, and $\texttt{Enc}(PK, a_2)$ to $\mathcal{C}$.

5.4 Making A Decision

$\mathcal{C}$ first decrypts $\texttt{Enc}(PK, \sum_{i=1}^{4} plain_i \cdot \theta_i)$ and determines whether the plaintext is 0. If not, $\mathcal{C}$ will reject. Otherwise $\mathcal{C}$ decrypts $\texttt{Enc}(PK, a_2)$, $\texttt{Enc}(PK, a_3)$ and $\texttt{Enc}(PK, a_2)$. Then, $\mathcal{C}$ determines whether $a_2 a_3 = a_4$. If so, he will accept that y is the correct solution of his computational task.

5.5 Security Analysis

THEOREM 2. *(Correctness) If the arithmetic circuit x is satisfiable, then a prover $\mathcal{P}$ with the knowledge of the correct input y is able to make the client $\mathcal{C}$ accept y by performing our protocol.*

PROOF. It is easy for a prover $\mathcal{P}$ who has found out the correct result y of the computation task to find out the correct assignment $z = (z_1, z_2, \cdots, z_n)$ including the correct result y and all the intermediate results of the circuit. That is, z satisfies all the constraints $f_u, u = 1, \cdots, n$. If $\mathcal{P}$ responds with correct values as in the protocol, all corresponding test equations (unencrypted version) will hold. Given the encryption used in this section is additive homomorphic, the ciphertexts of all the corresponding linear combinations are ciphertexts of 0. The conclusion follows. □

DEFINITION 3. *We say that a verification protocol for the arithmetic circuit satisfiability problem x wins* λ-confidentiality *if the following properties are satisfied. In the context of computational complexity, x can be represented as a binary string, and so can the correct assignment z. Let $P_x, P_z : \{0,1\}^* \to \{0,1\}$ be arbitrarily-defined predicates, extracting one bit of information about binary strings of arbitrary length. For every probabilistic polynomial-time algorithm A,*

for every possible transmitted messages $m = (m_1, \cdots, m_k)$ with $k = |m| = poly(\lambda)$, for every positive polynomial $p(\cdot)$, for every P_x, P_z, we have:

$$Pr[A(1^\lambda, m, 1^{|z|}, commt) = P_x(x))] - \frac{1}{2} < \frac{1}{p(\lambda)} \quad (5.1)$$

$$Pr[A(1^\lambda, m, 1^{|z|}, commt) = P_z(z))] - \frac{1}{2} < \frac{1}{p(\lambda)} \quad (5.2)$$

where commt is the commitment information provided by $\mathcal{C}$ before verification. (The probability is over z as well as over the internal coin tosses of either algorithms.)

THEOREM 3. *(Confidentiality) If the underlying homomorphic encryption in our protocol has the security parameter λ, then our verification protocol for the arithmetic circuit satisfiability problem wins λ-confidentiality.*

PROOF. Recall that the commitment is $commt = (\alpha_1, s_1, S_1, c_0, w_0 + \alpha_1 w_1', (r_1||R_1) + \alpha_0(r_0||R_0), \mathtt{Enc}(PK, (s_1 + S_1 + \alpha_0(s_0 + S_0)))$. Given that all transmitted messages $m = (m_1, m_2, \cdots, m_k)$ are encrypted in our protocol, for every probabilistic polynomial-time algorithm A there exists a probabilistic polynomial-time algorithm A^* such that

$$Pr[A(1^\lambda, m, 1^{|z|}, commt) = x_i)] < Pr[A^*(1^\lambda, 1^{|z|}, H) = x_i] + \frac{1}{p(\lambda)} \quad (5.3)$$

where $H = (\alpha_1, s_1, S_1, c_0, w_0 + \alpha_1 w_1', (r_1||R_1) + \alpha_0(r_0||R_0))$. This follows directly from an appropriate formulation of semantic security ([27], Def. 5.2.1) of the underlying homomorphic encryption. Now since $(r_0||R_0)$ is uniformly random, $(r_1||R_1) + \alpha_0(r_0||R_0))$ is random and independent of $(r_1||R_1)$ by the crypto lemma. Therefore, s_1, S_1 is the response of the prover to a query (on z) which is random and independent of $(\alpha_1, c_0, w_0 + \alpha_1 w_1', (r_1||R_1) + \alpha_0(r_0||R_0))$. This implies that the entire H is independent of x and z, and hence for any algorithm A^*, $Pr[A^*(1^\lambda, 1^{|z|}, H) = x_i] \leq \frac{1}{2} + \frac{1}{p(\lambda)}$ The inequalities in (5.1) and (5.2) follow. □

Theorem 3 implies that for sufficiently large λ, the advantage of an adversary finds out the circuit and the results in the execution of the protocol is negligible.

THEOREM 4. *(soundness) The client will accept a wrong answer with probability less than $\leq \frac{4}{|\mathbb{F}|} - \frac{1}{|\mathbb{F}|^2} + (3\delta - 6\delta^2)$ where the tests guarantee the proof is δ-close to linear.*

The proof is in Appendix of the full version paper [1].

6. PRACTICAL USE AND COMPLEXITY ANALYSIS

6.1 Amortized query costs

Our proposed schemes can amortize query costs. That is, our designs are able to use the same commitment/decommitment queries and PCP queries across many instances of the same circuit (with different input/output). The same commitment/decommitment queries and PCP queries make sure that it is still infeasible for $\mathcal{P}$ to know the secret commit query and provide uncommitted responses. (It is known that if we fix a given instance, the probability of wrongly accepting will not be influenced by other instances [42].)

The amortizing usage is as follows.

1. There is an off-line stage. In this stage, $\mathcal{C}$ reads the circuit and constructs the commitment queries and sends to $\mathcal{P}$. This is done only once.
2. $\mathcal{P}$ possesses β proofs (linear functions) $\tilde{\pi}_1, \cdots, \tilde{\pi}_\beta$, one for each instance. For commitment construction, $\mathcal{P}$ will return β tuples of commitments, each of which is as in the previous section.
3. For each instance, $\mathcal{P}$ computes the results.
4. In the verification phase, for the *same* query tuple $q_1, \cdots, q_4, t$ from $\mathcal{V}$, $\mathcal{P}$ will response β tuples of answers, each for one instance. For each instance, $\mathcal{V}$ verifies results as in the previous section. Totally, $\mathcal{V}$ runs β decommitments, β circuit tests, and β quadratic consistency tests, one for each instance.

We give a practical use example here. Suppose $\mathcal{C}$ has a large number of computational tasks. All these tasks can be reduced to an instance of the Circuit-SAT problem with the same circuit. Before using the Cloud server to do computing, $\mathcal{C}$ will generate the commitment queries: w_0, r_1, R_1, r_0, R_0. This is an off-line stage and it runs only once for all instances. The on-line stage is as in Section 5. First, $\mathcal{C}$ gives the computing tasks to $\mathcal{P}$. Secondly, $\mathcal{P}$ computes the tasks and gives back the results and the commitments to $\mathcal{C}$. After choosing a verifier $\mathcal{V}$, $\mathcal{C}$ outsources the verification to $\mathcal{V}$. $\mathcal{V}$ advises $\mathcal{C}$ to accept or reject.

6.2 Complexity Analysis

Our design meets the efficiency goal outlined in Section 2.3. As in [25], we are ignoring the time of the offline stage, since the cost of generating the commit/decommit queries can be amortized over many instances. We compare the computational cost and the communication cost of $\mathcal{C}$ between our protocol and other related work in Table 2. In this table, *Mult* and *Add* are the cost of multiplication and addition in $\mathbb{F}$. *RNG* is the cost of generating a random number in $\mathbb{F}$. *Oper* is the cost of the additive homomorphic operation. In our design, the computational and communication complexity of $\mathcal{C}$ for each instance is $O(m)$, (m is the length of results y) and is not dependent on the circuit size n. This is much more efficient than all current two-party verification schemes. We observe that $\mathcal{V}$ is also very efficient. *Even the cost of both $\mathcal{C}$ and $\mathcal{V}$ combined is less than the verifier's cost in the state-of-the-art argument systems that rely only on standard cryptographic assumptions.*

7. CONCLUSIONS AND FUTURE DIRECTIONS

In Cloud Computing, a client outsources computation to a more powerful server – the prover. To ensure the correctness of the results returned by the prover, the client has to perform a verification stage that is often tedious and expensive. In this work, we introduce the idea of delegation of verification in Cloud Computing. This natural approach relieves the client from performing the verification of the outsourced-computation results by outsourcing it to a third party – the verifier. We propose the first scheme that provides efficient outsourcing of the verification, while at the same time preserving the confidentiality of both the computational task and its result from untrusted verifiers. Given that the computational tasks are not limited to a specific

Table 2: Comparison of Complexity for Each Instance

		Computation	Communication
Our Basic Scheme	$\mathcal{C}$'s cost	0	$m + O(1)$
	$\mathcal{V}$'s cost	$m \cdot Mult + m \cdot Add + \frac{1}{\beta}[n^2 \cdot Mult + (4n) \cdot RNG]$	$\frac{1}{\beta}O(n^2)$
Our Full Solution	$\mathcal{C}$'s cost	$(m+1) \cdot Mult + m \cdot Add + Enc + 3Dec$	$m + O(1)$
	$\mathcal{V}$'s cost	$poly(\lambda) \cdot Oper + O(1) \cdot Mult + O(1) \cdot Add + \frac{1}{\beta}[n^2 \cdot Mult + (4n) \cdot RNG]$	$\frac{1}{\beta}O(n^2)$
Re-computing		$\geq [poly(n) \cdot Mult + poly(n) \cdot Add]$	0
NP-proof		$poly(n) \cdot Mult + poly(n) \cdot Add$	n
Linear PCP (Ishai et al.)		$poly(n) \cdot Mult + poly(n) \cdot Add$	$O(n^2)$
GINGER		$[(m+1) \cdot Mult + m \cdot Add] + \frac{1}{\beta}[poly(n) \cdot Mult + poly(n) \cdot Add + (n^2+2n) \cdot RNG]$	$\frac{1}{\beta}O((n+m)^2)$

computational problem, it appears at a first glance that the fully-homomorphic encryption is necessary for hiding the computational task. However, by means of combining a novel commitment protocol and the linear PCP system with only additive homomorphic encryption, our design enables a honest-but-curious third party to perform the bulk of the verification procedure, without gaining access to information about the original computational task or its result. We are currently investigating delegation of verification with a verifier who does not perform the protocol faithfully – a *curious and lazy verifier*.

8. ACKNOWLEDGMENTS

This work was partially supported by NSF under grants No. CNS-0644238 and CNS-0831470. We appreciate anonymous reviewers for valuable suggestions and comments.

9. REFERENCES

[1] `http://www.eng.iastate.edu/~guan/paper/delegation_of_verification.pdf`.

[2] S. Arora, C. Lund, R. Motwani, M. Sudan, and M. Szegedy. Proof verification and the hardness of approximation problems. *J. ACM*, 45(3):501–555, May 1998.

[3] S. Arora and S. Safra. Probabilistic checking of proofs; a new characterization of NP. In *Proceedings of the 33rd Annual Symposium on Foundations of Computer Science*, SFCS '92, pages 2–13, Washington, DC, USA, 1992. IEEE Computer Society.

[4] M. J. Atallah and K. B. Frikken. Securely outsourcing linear algebra computations. In *Proceedings of the 5th ACM Symposium on Information, Computer and Communications Security*, ASIACCS '10, pages 48–59, New York, NY, USA, 2010. ACM.

[5] L. Babai, L. Fortnow, L. A. Levin, and M. Szegedy. Checking computations in polylogarithmic time. In *Proceedings of the twenty-third annual ACM symposium on Theory of computing*, STOC '91, pages 21–32, New York, NY, USA, 1991. ACM.

[6] L. Babai, L. Fortnow, and C. Lund. Nondeterministic exponential time has two-prover interactive protocols. In *Proceedings of the 31st Annual Symposium on Foundations of Computer Science*, SFCS '90, pages 16–25 vol.1, Washington, DC, USA, 1990. IEEE.

[7] M. Ben-Or, S. Goldwasser, J. Kilian, and A. Wigderson. Multi-prover interactive proofs: how to remove intractability assumptions. In *Proceedings of the twentieth annual ACM symposium on Theory of computing*, STOC '88, pages 113–131, New York, NY, USA, 1988. ACM.

[8] E. Ben-Sasson, O. Goldreich, P. Harsha, M. Sudan, and S. Vadhan. Robust PCPs of proximity, shorter PCPs and applications to coding. In *Proceedings of the thirty-sixth annual ACM symposium on Theory of computing*, STOC '04, pages 1–10, New York, NY, USA, 2004. ACM.

[9] E. Ben-Sasson, O. Goldreich, P. Harsha, M. Sudan, and S. Vadhan. Short PCPs verifiable in polylogarithmic time. In *Proceedings of the 20th Annual IEEE Conference on Computational Complexity*, CCC '05, pages 120–134, Washington, DC, USA, 2005. IEEE Computer Society.

[10] E. Ben-Sasson and M. Sudan. Short PCPs with polylog query complexity. *SIAM J. Comput.*, 38(2):551–607, May 2008.

[11] S. Benabbas, R. Gennaro, and Y. Vahlis. Verifiable delegation of computation over large datasets. In *Proceedings of the 31st annual conference on Advances in cryptology*, CRYPTO'11, pages 111–131, Berlin, Heidelberg, 2011. Springer-Verlag.

[12] M. Blum and S. Kannan. Designing programs that check their work. *J. ACM*, 42(1):269–291, Jan. 1995.

[13] D. Boneh and D. M. Freeman. Homomorphic signatures for polynomial functions. In *Proceedings of the 30th Annual international conference on Theory and applications of cryptographic techniques: advances in cryptology*, EUROCRYPT'11, pages 149–168, Berlin, Heidelberg, 2011. Springer-Verlag.

[14] G. Brassard, D. Chaum, and C. Crépeau. Minimum disclosure proofs of knowledge. *J. Comput. Syst. Sci.*, 37(2):156–189, 1988.

[15] R. Canetti, B. Riva, and G. N. Rothblum. Practical delegation of computation using multiple servers. In *Proceedings of the 18th ACM conference on Computer and communications security*, CCS '11, pages 445–454, New York, NY, USA, 2011. ACM.

[16] R. Canetti, B. Riva, and G. N. Rothblum. Two 1-round protocols for delegation of computation. Cryptology ePrint Archive, Report 2011/518, 2011. `http://eprint.iacr.org/`.

[17] G. Cormode, M. Mitzenmacher, and J. Thaler. Practical verified computation with streaming interactive proofs. In *Proceedings of the 3rd Innovations in Theoretical Computer Science Conference*, ITCS '12, pages 90–112, New York, NY, USA, 2012. ACM.

[18] I. Dinur. The PCP theorem by gap amplification. *J. ACM*, 54(3), June 2007.

[19] I. Dinur, E. Fischer, G. Kindler, R. Raz, and S. Safra.

PCP characterizations of NP: Towards a polynomially small error probability. In *Proc. 31st ACM Symp. on Theory of Computing*, pages 29–40, 1999.

[20] I. Dinur and P. Harsha. Composition of low-error 2-query PCPs using decodable PCPs. In *50th Annual IEEE Symposium on Foundations of Computer Science, FOCS 2009, October 25-27, 2009, Atlanta, Georgia, USA*. IEEE Computer Society, 2009.

[21] I. Dinur and O. Meir. Derandomized parallel repetition of structured PCPs. In *IEEE Conference on Computational Complexity*. IEEE Computer Society, 2010.

[22] F. Ergun and S. R. Kumar. Approximate checking of polynomials and functional equations. In *Proceedings of the 37th Annual Symposium on Foundations of Computer Science*, FOCS '96, pages 592–, Washington, DC, USA, 1996. IEEE Computer Society.

[23] U. Feige and J. Kilian. Making games short (extended abstract). In *STOC*, pages 506–516, 1997.

[24] M. Garofalakis. Proof sketches: Verifiable in-network aggregation. In *IEEE Internation Conference on Data Engineering (ICDE)*, 2007.

[25] R. Gennaro, C. Gentry, and B. Parno. Non-interactive verifiable computing: outsourcing computation to untrusted workers. In *Proceedings of the 30th annual conference on Advances in cryptology*, CRYPTO'10, pages 465–482, Berlin, Heidelberg, 2010. Springer-Verlag.

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

[27] O. Goldreich. *Foundations of Cryptography: Basic Applications*, page 381. Cambridge University Press, 2004.

[28] S. Goldwasser, Y. T. Kalai, and G. N. Rothblum. Delegating computation: interactive proofs for muggles. In *Proceedings of the 40th annual ACM symposium on Theory of computing*, STOC '08, pages 113–122, New York, NY, USA, 2008. ACM.

[29] S. Goldwasser and S. Micali. Probabilistic encryption & how to play mental poker keeping secret all partial information. In *Proceedings of the fourteenth annual ACM symposium on Theory of computing*, STOC '82, pages 365–377, New York, NY, USA, 1982. ACM.

[30] S. Goldwasser, S. Micali, and C. Rackoff. The knowledge complexity of interactive proof systems. *SIAM J. Comput.*, 18(1):186–208, Feb. 1989.

[31] P. Golle and I. Mironov. Uncheatable distributed computations. In *Proceedings of the 2001 Conference on Topics in Cryptology: The Cryptographer's Track at RSA*, CT-RSA 2001, pages 425–440, London, UK, UK, 2001. Springer-Verlag.

[32] Y. Ishai, E. Kushilevitz, and R. Ostrovsky. Efficient arguments without short PCPs. In *Proceedings of the Twenty-Second Annual IEEE Conference on Computational Complexity*, CCC '07, pages 278–291, Washington, DC, USA, 2007. IEEE Computer Society.

[33] G. O. Karame, M. Strasser, and S. Čapkun. Secure remote execution of sequential computations. In *Proceedings of the 11th international conference on Information and Communications Security*, ICICS'09, pages 181–197, Berlin, Heidelberg, 2009. Springer-Verlag.

[34] J. Kilian. A note on efficient zero-knowledge proofs and arguments (extended abstract). In *Proceedings of the twenty-fourth annual ACM symposium on Theory of computing*, STOC '92, pages 723–732, New York, NY, USA, 1992. ACM.

[35] J. Kilian. Improved efficient arguments (preliminary version). In *Proceedings of the 15th Annual International Cryptology Conference on Advances in Cryptology*, CRYPTO '95, pages 311–324, London, UK, UK, 1995. Springer-Verlag.

[36] O. Meir. Combinatorial PCPs with efficient verifiers. In *Proceedings of the 2009 50th Annual IEEE Symposium on Foundations of Computer Science*, FOCS '09, pages 463–471, Washington, DC, USA, 2009. IEEE Computer Society.

[37] S. Micali. Computationally sound proofs. *SIAM J. Comput.*, 30(4):1253–1298, Oct. 2000.

[38] D. Moshkovitz and R. Raz. Two-query PCP with subconstant error. *J. ACM*, 57(5):29:1–29:29, June 2008.

[39] A. Polishchuk and D. A. Spielman. Nearly-linear size holographic proofs. In *Proceedings of the twenty-sixth annual ACM symposium on Theory of computing*, STOC '94, pages 194–203, New York, NY, USA, 1994. ACM.

[40] B. Przydatek, D. Song, and A. Perrig. Sia: secure information aggregation in sensor networks. In *Proceedings of the 1st international conference on Embedded networked sensor systems*, SenSys '03, pages 255–265, New York, NY, USA, 2003. ACM.

[41] S. Setty, A. J. Blumberg, and M. Walfish. Toward practical and unconditional verification of remote computations. In *Proceedings of the 13th USENIX conference on Hot topics in operating systems*, HotOS'13, pages 29–29, Berkeley, CA, USA, 2011. USENIX Association.

[42] S. Setty, R. McPherson, A. J. Blumberg, and M. Walfish. Making argument systems for outsourced computation practical (sometimes). In *NDSS*, 2012.

[43] S. Setty, V. Vu, N. Panpalia, B. Braun, A. J. Blumberg, and M. Walfish. Taking proof-based verified computation a few steps closer to practicality. In *USENIX Security*, 2012.

[44] R. Sion. Query execution assurance for outsourced databases. In *Proceedings of the 31st international conference on Very large data bases*, VLDB '05, pages 601–612. VLDB Endowment, 2005.

[45] B. Thompson, S. Haber, W. G. Horne, T. Sander, and D. Yao. Privacy-preserving computation and verification of aggregate queries on outsourced databases. In *Proceedings of the 9th International Symposium on Privacy Enhancing Technologies*, pages 185–201, Berlin, Heidelberg, 2009. Springer-Verlag.

[46] C. Wang, K. Ren, and J. Wang. Secure and practical outsourcing of linear programming in cloud computing. In *INFOCOM*, pages 820–828. IEEE, 2011.

[47] C. Wang, K. Ren, J. Wang, K. Mahendra, and R. Urs. Harnessing the cloud for securely solving large-scale systems of linear equations. In *ICDCS*, pages 549–558. IEEE, 2011.