INVITED PAPER *Special Issue on Info-Plosion*

Execution Assurance for Massive Computing Tasks

Ting WANG[†] ***and*** **Ling LIU**[†]**,**

SUMMARY

Consider a client who intends to perform a massive computing task comprsing a number of sub-tasks, while both storage and computation are outsourced by a third-party service provider. How could the client ensure the integrity and completeness of the computation result? Meanwhile, how could the assurance mechanism incur no disincentive, e.g., excessive communication cost, for any service provider or client to participate in such a scheme? We detail this problem and present a general model of execution assurance for massive computing tasks. A series of key features distinguish our work from existing ones: a) we consider the context wherein both storage and computation are provided by untrusted third parties, and client has no data possession; b) we propose a simple yet effective assurance model based on a novel integration of the machineries of data authentication and computational private information retrieval (cPIR); c) we conduct an analytical study on the inherent trade-offs among the verification accuracy, and the computation, storage, and communication costs.

key words: *Execution assurance, massive computing, computational private information retrieval*

1. Introduction

Spurred by the unprecedented data growth rate [1] and the high cost of maintaining enterprise information infrastructures, companies and business organizations are embracing the paradigm of storage and computation outsourcing wherein third parties (service providers) host and manage their customers' business data, and support online computation for end clients. More specifically, cloud computing, exemplified by Amazon's Elastic Compute Cloud (EC2) [2], Microsoft's Azure Service Platform [3], and Rackspace's Mosso [4], is envisoned to provide the next generation of infrastructure support for online computation outsourcing.

In addition to its advantages (e.g., economies of scale, dynamic provisioning, and low capital expenditures), such frameworks also introduce a range of new risks, including 1) the communication-layer security, 2) the privacy of both data and access patterns, and 3) the integrity of computation output. In this paper, we specifically focus on the third aspect of security risks; that is, the server could be compromised by malicious adversaries, and deceiving clients might be attempted to gain business advantages, or for competition against important clients, or for easing the workload on the server, among other motives.

Fig. 1 illustrates the framework of storage and computation outsourcing. The data owner deposits its business

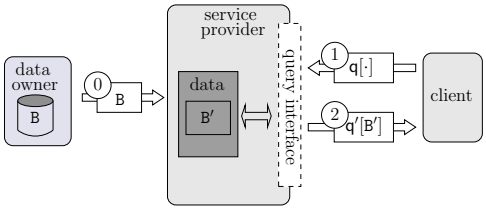

Fig. 1 Model of storage and computation outsourcing.

data B at a third-party service provider (Phase 0) that is responsible for providing online computation services for end clients, e.g., executing task request q (Phase 1), and delivering results q[B] to the clients (Phase 2). The service provider might attempt malicious deception by replacing the original data B or request q with tampered version B′ or q′, respectively. The goal of execution assurance is to detect such deception by ensuring the correctness of the executed computation, i.e., validating the equivalence of q′[B′] and q[B].

1.1 State of The Art

While offering great business opportunities [5], the database-as-service paradigm (DAS) also raises severe privacy and security concerns. The problem of providing execution assurance under the outsourcing model has been addressed to certain limited extent in both security and database communities. The existing solutions can be roughly classified into three categories.

The first one focuses on ensuring the integrity and completeness of the retrieved (entire or partial) source data [6] [7] [8], i.e., the equivalence of B and B′ in Fig. 1, or the proof of its retrievability [9] [10] [11].

The second category of work [12] [13] [14] [15] [16] [17] aims at providing assurance for queries with logical comparison predicates, i.e., q is limited to queries that select relational tuples matching simple predicates, e.g., range queries. According to their backbone techniques, the existing solutions can be categorized as Merkle hash tree (MHT)-based approaches [12] [15] [16], or signature-chaining-based ones [17] [14] [13]. Ensuring integrity and completeness for general computation tasks, however, remains a difficult open issue.

The third direction of research efforts concentrate on achieving assurance from the data owner side [18] [19], i.e., the client has access to the entire collection of B in Fig. 1. Targeting lazy adversaries, [18] focuses on ensuring the actual execution of the computation at the server, via an ex-

Manuscript received January 1, 2010.
Manuscript revised January 1, 2010.

[†]{twang, lingliu}@cc.gatech.edu, Distributed Data Intensive Systems Lab, Georgia Institute of Technology, USA

DOI: 10.1587/trans.E0.??.1

tended version of the ringer scheme [20]. It however does not offer protection against malicious adversary who intensionally tampers the computation result before sending it back to the client. [19] considers the problem of verifying the correctness of aggregation and selection queries over data streams using polynomial identity random synopses.

Nevertheless, none of the approaches above are applicable to our setting, given their requriement for data possession by the client. As an equally important scenario, enforcing assurance from end clients who have no possession of source data remains a challenging problem.

1.2 Scope and Problem Definition

In this paper, we intend to address the problem of providing execution assurance for massive computing tasks while both storage and computation are provided by untrusted service providers. Specifically, we target the setting wherein the computation task comprises a number of sub-tasks, each as a black-box function; 2) the end client (the task issuer) has no possession of the original input data. Under this setting, assurance is needed to ensure that 1) the input data is authenticate, 2) the batch of sub-tasks are completely executed according to the Service Level Agreement (SLA), and 3) each sub-task is executed correctly over the intended portion of input data.

Evidently, a straightforward solution to bypassing the difficulty is to provide the clients with direct access to the original data $\mathtt{B}$ by the data owner or other authorities; for this transformed problem, a bulk of existing solutions are available. The feasibility of this naïve solution, however, is questionable: 1) the data owner might have insufficient information infrastructure to support clients' requests, which is essentially one main motive of data management outsourcing; 2) the trustworthiness of commercial data storage providers is usually problematic in real applications. Therefore, one needs to consider execution assurance for massive computing tasks without data possession as a unique problem demanding specifically designed solution.

1.3 Contributions

This work presents a general model for validating the integrity and completeness of the results of massive computing tasks. The overall framework is constructed atop a client-side random checking mechanism: the client creates a set of *challenge objects* based on a set of randomly selected sub-tasks and a small fraction of input data retrieved from the server. The returned result is evaulated against the challenge objects, and their equivalence functions as the prerequisite to accept the result as accurate.

More specifically, our solution is built partially based on the theory of computational *private information retrieval* (cPIR) [21], which allows a client to privately retrieve information from a server without exposing the requested information. By carefully integrating cPIR with the cryptographic machinery of data authentication, an end client is enabled to construct the challenge objects requiring no direct access to the original input data. Furthermore, the task result checking is achieved by an efficient sampling scheme, which provides probabilistic guarantee of verification accuracy. Although cPIR has been criticized of being excessively computationally costly to be practical [22], for given verificaiton accuracy, our scheme requires to retrieve only a small, constant amount of data, independent of the size of entire data collection, which implies the feasibility of cPIR in our framework.

Our contributions can be summarized as follows:

- Identity and articulate the problem of execution assurance for oursourced massive computing tasks without data possession, and expose its intrinsic characteristics and technical challenges;
- Propose an efficient client-side solution via novel integration of the machineries of data authentication and computational private information retrieval;
- Provide a theoretical analysis regarding the intricate trade-offs among the signature construction cost (by data owner), the communication and computation costs (by server and client), and the verification accuracy.

2. Models and Assumptions

2.1 Essentials

The framework of massive computing outsourcing follows the basic model as shown in Fig. 1, with three main entities as data owner, service provider, and end client. In this paper, we consider the case that the data owner and the end client exist independently; hence, the client has no direct access to the original data (without data possession). Following, we clarify a set of fundamental concepts.

Definition 1 (Data): The input data $\mathtt{B}$ comprises n data blocks $\mathtt{B} = \{\mathtt{b}_i\}_{i=1}^{n}$. A block is the basic unit over which the computation is executed.

We assume that the number of blocks n is known to all the entities, and the client has the privilege of accessing all the blocks. The imposition of access control and trust management [23] [24] for both client and server is perpendicular to the focus of this work.

Definition 2 (Task): A massive computing task $\mathtt{Q}$ consists of a set of sub-tasks $\mathtt{Q} = \{\mathtt{q}_i\}_{i=1}^{m}$. Each sub-task $\mathtt{q}_i$ can be considered as a black-box function that takes as input data $\mathtt{B}$, and produces output $\mathtt{q}_i[\mathtt{B}]$.

Without loss of generality, we assume that $m = n$, and each sub-task $\mathtt{q}_i$ is only executed over the data block $\mathtt{b}_i$; thus, we can use the index i to refer to both the sub-task $\mathtt{q}_i$ and the block $\mathtt{b}_i$. For a sub-task that takes as input multiple data blocks from the database $\mathtt{B}$, one can typically decompose it into even smaller sub-tasks that take single blocks as input. We note that this definition is fairly general in the sense that it is only required to be a valid surjective mapping: given the

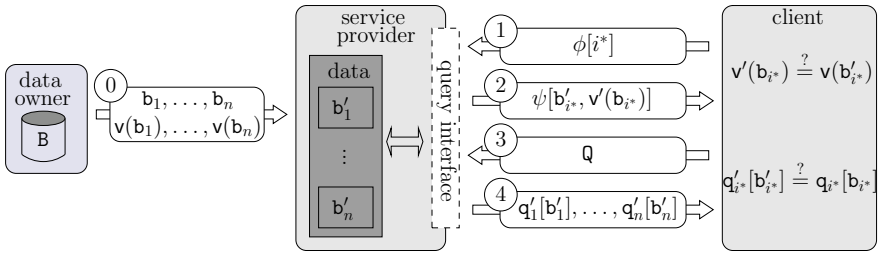

Fig. 2 Framework of execution assurance for massive computing tasks.

same input, it generates identical output, i.e., $\mathtt{q}_i[\mathtt{b}_i] = \mathtt{q}_i[\mathtt{b}'_i]$ if $\mathtt{b}_i = \mathtt{b}'_i$.

Furthermore, we assume that the client lacks sufficient computation capability to execute the entire computing task Q, but is equipped to 1) verify the authenticity of data blocks using digital signatures and 2) execute a limited number of sub-tasks if given the input data.

2.2 Adversary Model

In general, the untrusted service provider [†] may compromise the integrity and completeness of computation output in two ways: 1) without actually executing a sub-task q over an block b, she guesses the query result q[b]′; or 2) after executing q over b and obtaining the correct result q[b], she replaces it with a modified version q[b]′. Although caused by different behaviors, the faked or tampered computation results are essentially equivalent, from the perspective of end client. Hence we argue that it is possible to design a unified execution assurance solution against both faking and tampering attacks. It is worth emphasizing that the techniques designed for ensuring execution completeness, e.g., [18], can be readily plugged into our framework to help achieve even stronger security. In this paper, we design our solution aiming at a general adversary model as follows:

Definition 3 (Adversary): Given a computing task $\mathtt{Q} = \{\mathtt{q}_i\}_{i=1}^n$ and a collection of input data $\mathtt{B} = \{\mathtt{b}_i\}_{i=1}^n$, among the n results $\{\mathtt{q}_i[\mathtt{b}_i]\}_{i=1}^n$ delivered to the client, the adversary compromises e of them, while keeping the rest $(n-e)$ ones intact.

2.3 Basic Building Blocks

We now introduce the basic blocks that serve as the foundation of our execution assurance model.

Definition 4 (Hash): A hash function $\mathcal{H}(\cdot)$ is an efficiently computable function that takes a variable-length input x and generates a fixed-length output $\mathtt{y} = \mathcal{H}(\mathtt{x})$. The hash functions used in this work are assumed to be collision resistant, i.e., it is computationally infeasible to find two inputs $\mathtt{x} \neq \mathtt{x}'$ such that $\mathcal{H}(\mathtt{x}) = \mathcal{H}(\mathtt{x}')$.

One example of such collision-resistant hash function is SHA1 [25], which takes variable-length bit-string and produces 160-bit output.

[†]Henceforth, without ambiguity, we use the terms "adversary" and "server" interchangeably.

Meanwhile, a public-key digital signature scheme enables authenticating the integrity of a signed message [26]:

Definition 5 (Signature): After constructing a pair of keys (sk, pk), the signer keeps sk secret, and publishes the public key pk. Subsequently, for any message x sent by the signer, an unforgeable signature $\mathtt{sig}_{\mathtt{sk}}(\mathtt{x}) = \mathcal{S}(\mathtt{x}, \mathtt{sk})$ is produced. The recipient can verify the integrity of x by running a verification procedure $\mathcal{V}(\mathtt{x}, \mathtt{pk}, \mathtt{sig}_{\mathtt{sk}}(\mathtt{x}))$.

3. Execution Assurance: Our Solution

This section presents our framework of providing execution assurance without data possession, and elaborates on the key techniques involved in building the solution.

3.1 Overview

Fig. 2 sketches the overall framework of our computing assurance solution. The three entities involved in the play are data owner, service provider, and end client. In the setup stage (step 0), the data owner outsources its business data as a collection of blocks $\mathtt{B} = \{\mathtt{b}_i\}_{i=1}^n$ to the service provider. Each block b is associated with an unforgeable *verification object* v(b) signed by the data owner, which enables the client to verify the authenticity and ownership of the data retrieved from the service provider.

The assurance of computation execution is achieved in two phases, *challenge objects contruction* and *computation result verification*. Specifically, in the first phase (step 1 and 2 in Fig. 2), the client randomly samples a set of data blocks, $\{\mathtt{b}_i^s\}_{i=1}^f$, from the entire collection at the service provider. After verifying their authenticity via the associated verification objects, the client composes a set of challenge objects $\{\mathtt{c}(\mathtt{q}_i^s)\}_{i=1}^f$, corresponding to the sampled blocks. In the second phase (step 3 and 4 in Fig. 2), after receiving the results returned by the service provider, $\{\mathtt{q}_i[\mathtt{b}_i]'\}_{i=1}^n$, the end client validates the results corresponding to the sampled set $\{\mathtt{q}_i^s\}_{i=1}^f$ using the constructed challenge objects. The batch of computation results are accepted as accurate, only if all the verifications indicate positive.

Before presenting the detailed techniques involved in each phase, we first demonstrate the probablistic guarantee provided by our verification scheme. Following the assumption that the adversary compromises e out of n sub-task results, and the client tests f of the received results independently at random, the probability that the adversary is de-

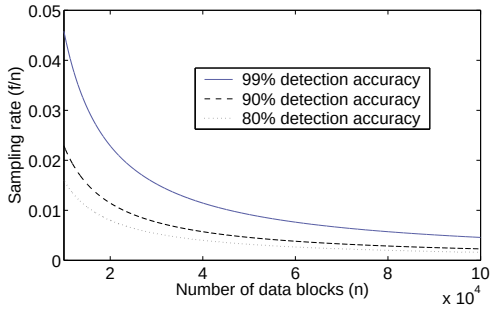

Fig. 3 Sampling rate with respect to the size of data collection and the required counterfeit detection accuracy.

tected of performing counterfeit, $P(n, e, f)$, is equivalent to the classical sampling experiment without replacement:

$$P(n, e, f) = 1 - \frac{C_{n-e}^{f}}{C_{n}^{f}} \tag{1}$$

It can be derived that if e/n is a constant fraction, e.g., 1%, the minimum number of samples (denoted as t^*) required to achieve certain level of counterfeit detection accuracy λ, i.e., $P(n, e, t^*) \geq \lambda$, is constant and independent of n. In particular, since $\frac{n-e-i}{n-i} < \frac{n-e}{n}$ for any $i > 0$, for given λ, we have the following upper bound for t^*:

$$t^* < \lceil \log_{\frac{n-e}{n}}(1-\lambda) \rceil$$

Fig. 3 shows the required sampling rate t^*/n with respect to the size n of the entire data collection which varies from 10^4 to 10^5, and $e/n = 0.01$, when the counterfeit detection accuracy λ is set as 0.8, 0.9, and 0.99. It is noticed that 1) the required sampling rate t^*/n is fairly low, even for very high detection accuracy, e.g., t^*/n is below 1% for $\lambda = 99\%$ when n reaches 5×10^4; 2) for given detection accuracy λ, the required sampling rate t^*/n decreases reciprocally as n grows, which implies approximately constant costs of network communication and computation.

Clearly, the key element leading to the effectiveness of our execution assurance scheme lies in the ability of the client to privately construct the set of challenge objects, which in turn depends on the retrieval of untampered data samples from the service provider. In the following, we proceed to elaborating on how our approach achieves this goal.

3.2 Private Data Sampling and Authentication

A straightforward, yet unrealistic solution to achieving private sampling of f out of n blocks would be to transfer the entire collection of input data to the client side, which however incurs the prohibitive communication cost of $O(n)$, i.e., linear in the size of the entire data collection.

We, instead, construct our solution partially based on computational private information retrieval (cPIR), which allows a client to privately retrieve information from a server without exposing the particular requested information. For ease of presenation, in this paper we exemplify the cPIR schemes based on group-homomorphic encryption [21], while our framework is general enough to be built on other PIR schemes, e.g., the one constructed based on ϕ-hiding number-theoretic assumption [27].

symbol	definition
$\mathtt{B}$	data collection
$\mathtt{b}$	data block
n	number of blocks in $\mathtt{B}$
$\mathtt{Q}$	massive computing task
$\mathtt{q}$	sub-task
$\mathcal{H}(\cdot)$	hash function
$\mathcal{S}(\cdot)$	signing function
$\mathcal{V}(\cdot)$	verifying function
$\mathtt{h}$	hash digest
$\mathtt{sig}$	signature digest
$\mathsf{v}(\cdot)$	verification object
$\mathsf{c}(\cdot)$	challenge object
e	number of counterfeits
f	number of challenge objects

Table 1 Symbols and notations.

3.2.1 Sampling with Basic cPIR

The database $\mathtt{B}$ is organized into a $r \times c$ matrix ($n = r \times c$) such that each block $\mathtt{b}$ is mapped to one unique position of the matrix. Note that the placement of the blocks is flexible, given that it is agreed upon among the data owner, the server and the client. Following, we denote the data collection as $\mathtt{B} = \{\mathtt{b}_{ij}\}$ ($1 \leq i \leq r, 1 \leq j \leq c$).

Each entry of the matrix corresponds to the concatenation of the data block $\mathtt{b}$, and its verification object $\mathsf{v}(\mathtt{b})$, i.e., $\mathtt{b}||\mathsf{v}(\mathtt{b})$. Let d be the length of the data entry: $d = \max_{i,j} |\mathtt{b}_{ij}||\mathsf{v}(\mathtt{b}_{ij})|$. The database at the service provider consists of d matrices $\{M_k\}_{k=1}^{d}$ with M_k comprised of the k-th bit of each entry. Without loss of generality, the following discussion considers the case of $d = 1$.

Let $(\mathcal{K}, \mathcal{E}, \mathcal{D})$ ($\mathcal{K}$, $\mathcal{E}$ and $\mathcal{D}$ represent the key generation, encryption, and decryption algorithms, respectively) be a cryptosystem which satisfies 1) secure against chosen plaintext attacks, and 2) homomorphic over an abelian group G, i.e., given the plaintext and ciphertext sets as G and G', $\mathcal{D}(\mathcal{E}(a) \times \mathcal{E}(b)) = a * b$ where $a, b \in G$, and $\times$, $*$ represent the group operations of G and G', respectively.

Assuming that the client intends to retrieve $\mathtt{b}_{i^*j^*}$ of the database and its associated verification object, the cPIR scheme works as follows:

- The client picks a non-identity element $g \in G$, and sends a query of the form $X = \{x_j\}_{j=1}^{c}$ such that

$$\mathcal{D}(x_j) = \begin{cases} \text{identity of } G & j \neq j^* \\ g & j = j^* \end{cases} \tag{2}$$

- The server responds with a set of results $Y = \{y_i\}_{i=1}^{r}$, where y_i is defined as $\prod_{j=1}^{c} \mathtt{b}_{ij} \cdot x_j$. Without ambiguity, we use the multiplicative notation to denote the operations of both G and G', and use $\cdot$ to represent the $\mathbb{Z}$ module operation;
- After receiving Y, the client can now recover the intended entry $\mathtt{b}_{i^*j^*}$ via computing the quantity $\mathcal{D}(y_{i^*})$ as below: $\mathcal{D}(y_{i^*}) = \mathcal{D}\left(\prod_{j=1}^{c} \mathtt{b}_{i^*j} \cdot x_j\right) = \mathtt{b}_{i^*j^*} \cdot \mathcal{D}(x_{j^*}) = \mathtt{b}_{i^*j^*} \cdot g$, i.e., $\mathcal{D}(y_{i^*}) = g$ only if $\mathtt{b}_{i^*j^*} = 1$.

Assume that each $x \in X$ and $y \in Y$ are encoded in a same length, the communication complexity of this cPIR scheme is $O(c + d \cdot r)$ (for sending query X and delivering d sets of Ys), and the computation at the server mainly involves $d \cdot c \cdot r$ times of group operations. Therefore, for given n and d, the overall communication complexity is minimized when $r = \sqrt{n/d}$ and $c = \sqrt{n \cdot d}$. We will revise this setting when the construction of verification objects are taken into consideration.

3.2.2 Verification Object

The verification object $\mathsf{v}(\mathsf{b})$ of a data block b enables a client to verify its authenticity, which is the prerequisite for constructing a valid challenge object. Our design of verification object aims to minimize 1) its size, since as revealed above, the communication and computation costs linearly depend on the length of each data entry $|\mathsf{b}||\mathsf{v}(\mathsf{b})|$; and 2) its construction cost (for the data owner), which is especially important in the case of frequent data updates. Without loss of generality, below, we assume that $n = 2^d$ and $r = c = \sqrt{n}$.

A straightforward solution to verifying n blocks is to construct a Merkle hash tree (MHT) [28], a method of collectively authenticating a set of objects. Specifically, in our setting, one can adopt a quadrant variant (q-MHT), which is essentially a 4-way balanced tree structure: each leaf node corresponds to the hash (digest) of a data entry, each non-leaf node corresponds to the hash of the concatenation of its direct children, and the root contains the digital signature of its digest. The verification object of each block consists of the set of hash digests of the sibling nodes along its path to the root, which enables the client to reconstruct the root of MHT, and verify the integrity by comparing its equivalence with the original signature.

In implementation, the entire data block matrix can be divided into $2^{d-d'}$ equal-size squares, called data squares, and within each square, a q-MHT of height $d'/2$ is constructed. The verification object of each data block is therefore of size $\frac{3}{2}d'|\mathsf{h}| + |\mathtt{sig}|$, and totally $2^{d-d'}$ signatures are required for the data owner. Clearly, a trade-off exists between the communication and signature construction costs: at one extreme, $d' = d$, the whole database is considered as a data square, which achieves the minimum signature construction cost, at the expense of the largest size of verification object; while at the other extreme, $d' = 0$, each block itself is a data square, with the minimum communication cost, but the largest number of required signing operations.

We have the following key observation regarding the property of the cPIR scheme: at each round the server returns the encrypted version of an entire column, instead of a single data block. Based on this observation, we propose a novel *vertical Merkel hash tree* (v-MHT), a variant of MHT that achieves minimum verification object size and minimum construction cost simultaneously.

For each data column j, one computes its digest $\mathsf{h}_{\cdot,j}$ as the hash of the concatenation of the digests of its elements: $\mathsf{h}_{\cdot,j} = \mathcal{H}(\bigsqcup_{i=1}^{r} \mathsf{h}_{i,j})$, where $\bigsqcup$ denotes the concatenation operation. The root corresponds to the hash digest of the concatenation of the column digests, and a signature $\mathcal{S}(\mathcal{H}(\bigsqcup_{j=1}^{c} \mathsf{h}_{\cdot,j}))$ is created. For the j^*-th column, one constructs a meta verification object $\mathsf{v}(\mathsf{b}_{\cdot,j^*})$ as the concatenation of the digests of all but the j^*-th column and the signature at the root: $\mathsf{v}(\mathsf{b}_{\cdot,j^*}) = \bigsqcup_{j=1, j\neq j^*}^{c} \mathsf{h}_{\cdot,j} \| \mathtt{sig}$. One then divides $\mathsf{v}(\mathsf{b}_{\cdot,j^*})$ into r equal-length parts, denoted as $\{\mathsf{v}_i(\mathsf{b}_{\cdot,j^*})\}_{i=1}^{r}$, with $\mathsf{v}_{i^*}(\mathsf{b}_{\cdot,j^*})$ assigned as the verification object of $\mathsf{b}_{i^*j^*}$. A sample v-MHT is illustrated in Fig. 4. On receiving the verification objects of an entire column, the client can readily re-construct the root of v-MHT, and validate the data integrity in sequel. Clearly, within this scheme, the verification object of each data entry is of size $[(c-1)\cdot|\mathsf{h}| + |\mathtt{sig}|]/r$ ($[(\sqrt{n}-1)\cdot|\mathsf{h}| + |\mathtt{sig}|]/\sqrt{n}$ for the case of $r = c$), and only one signing operation is required for the data owner. Also, it can be derived that under this setting, in order to minimize the communication cost, the optimal setting of c, r is

$$c^* = \sqrt{n \cdot |\mathsf{b}|/(|\mathsf{h}| + 1)} \qquad r^* = n/c^* \tag{3}$$

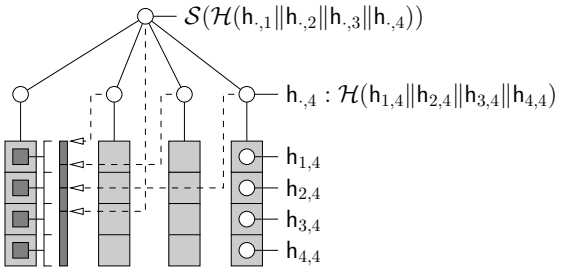

Fig. 4 Illustration of vertical Merkle hash tree.

3.3 Computation Execution and Verification

On obtaining the set of sampled input data blocks and ensuring their authenticity, the client enters the second phase: she constructs a set of challenge objects, which later serve to validate the integrity and completeness of the received computation result. In this section, we discuss in detail the construction of challenge objects, and model the intricate trade-off between the construction cost and the corresponding verification power, with respect to several alternative attack strategies.

3.3.1 Construction of Challenge Object

In general, two criteria need to be considered in execution assurance, namely, completeness and integrity. By completeness, we refer to that all the given sub-tasks have been faithfully executed, and a server violating this criterion is termed as a *lazy adversary*. By integrity, we refer to that the computation results have not been tampered or modified before being sent to the client, and a server violating this criterion is termed as a *malicious adversary*. This work mainly focuses on the case of malicious adversaries, with the aim of ensuring the integrity of computation results.

Clearly, given the result $\mathsf{q}[\mathsf{b}]'$ as returned by the service provider for the sub-task q over the block b, the client

can verify its integrity only if she knows the correct $\mathtt{q}[\mathtt{b}]$, by checking the equivalence of $\mathtt{q}[\mathtt{b}]$ and $\mathtt{q}[\mathtt{b}]'$. Hence, each challenge object $\mathtt{c}(\mathtt{q})$ is essentially the "correct" result of performing the sub-task $\mathtt{q}$ over the targeted data block $\mathtt{b}$. Moreover, it usually suffices to check the equivalence of their hash digests to determine the equivalence of $\mathtt{q}[\mathtt{b}]$ and $\mathtt{q}[\mathtt{b}]'$, leading to lower storage requirement. We thus have

Definition 6 (Challenge Object): For a sampled data block b_i^s with verified authenticity, a challenge object $\mathtt{c}(\mathtt{q}_i^s)$ is constructed as the hash digest of the result of performing the sub-task $\mathtt{q}_i^s$ over $\mathtt{b}_i^s$, i.e., $\mathtt{c}(\mathtt{q}_i^s) = \mathcal{H}(\mathtt{q}_i^s[\mathtt{b}_i^s])$.

Provided the collision resistance of the hash function, it is computationally infeasible to fake a result $\mathtt{q}[\mathtt{b}]'$ with an equivalent hash digest of $\mathtt{q}[\mathtt{b}]$.

3.3.2 Amortization of Construction Cost

The discussion so far has been assuming that at each round of random sampling, a single challenge object is constructed. Thus, all the challenge objects are constructed independently at random, which offers the maximum possible verification power, according to Eq. 1, but also incurs the prohibitive challenge-object construction cost at the client side. Specifically, the cost comprises two main parts, the transfer of the data blocks and the associated verification objects, and the construction of the set of challenge objects. Following, we show that the inherent structure of the random sampling scheme allows flexible balance between the construction cost and the corresponding verification power.

It is noticed that for each round of communication, the random sampling scheme retrieves a complete column of data blocks of the database. Therefore, instead of constructing a single challenge object, one can construct multiple ones from the retrieved column, to save the network bandwidth. However, the amelioration on communication cost is achieved at the expense of breaking the ideal assumption that all the challenge objects are constructed independently at random, since now it is known to the adversary that the set of challenge objects are constructed from the data entries belonging to a single column. Clearly, a balance exists between the cost of constructing the challenge objects and their verification power, about which a theoretical analysis is presented in the following.

More generally, assume that a set of g columns $\{B_i^s\}_{i=1}^g$ of the data collection $\mathtt{B}$ are retrieved, each representing a distinct column of the collection. The end client can construct the challenge objects from a pool of $r \cdot g$ data blocks. Meanwhile, note that two challenge objects corresponding to an identical data block $\mathtt{b}$ does not provide higher verification power than each single one, i.e., the adversary can pass the verification by faithfully returning the correct result $\mathtt{q}[\mathtt{b}]$. Hence, we assume that all f challenge objects are constructed from distinct data blocks. Aiming at addressing the worst-case scenario, we further assume that the numbers (g, f) both are known to the adversary. The relationships between the communication cost Cost_{comm}, the challenge object construction cost Cost_{cons}, and the parameters (g, f) can be summarized as $\mathrm{Cost}_{comm} \propto g$, $\mathrm{Cost}_{cons} \propto f$. Below we analyze the verification power of the challenge objects with respect to several alternative attack strategies exploitable by the adversary.

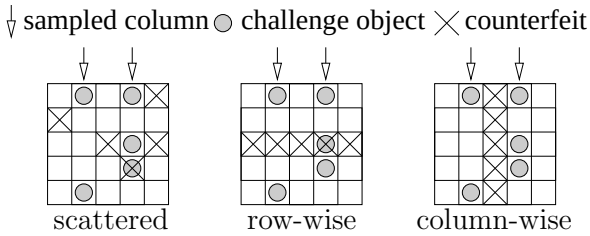

Fig. 5 Alternative counterfeit strategies of the adversary.

3.3.3 Analysis of Verification Power

As revealed above, the amelioration on the communication and challenge object construction costs is achieved by introducing the correlations among the challenge objects with respect to their corresponding data blocks. The adversary can potentially leverage such correlations to improve her chance of escaping from the integrity verification, given the same amount of counterfeits in the query results.

Following the adversary model defined in Section 2.2, we assume that the adversary attempts to counterfeit e out of n sub-tasks of a massive computing task $\mathtt{Q}$. We also assume that compromising the result corresponding to every sub-task offers the same amount of incentive for the adversary. Though this assumption is somewhat restrictive, the analysis presented below provides interesting insights into the probabilistic behavior of the verification power of our scheme. We consider three alternative attack strategies used by the adversary in choosing the sub-tasks to inject compromised results, namely *scattered*, *column-wise*, and *row-wise* counterfeit, and analyze their impacts over the verification power of the set of challenge objects.

Scattered Counterfeit. Within this basic scheme, the set of e compromised sub-tasks are chosen randomly at the basis of individual sub-task, as illustrated in the leftmost plot of Fig. 5. The probability that the adversary manages to "dodge" the f challenge objects embedded in the g columns can be calculated as follows:

$$P_{scattered}(e|f,g,n) = \frac{C_{n-f}^e}{C_n^e}$$

which is equivalent to the ideal case as shown in Eq. 1, where all the challenge objects are constructed independently at random, i.e., the correlation does not lend the scattered counterfeit scheme any advantage in dodging the challenge objects. Therefore, this scheme will serve as the benchmark for evaluating other counterfeit strategies.

Row-wise Counterfeit. In the strategy of row-wise counterfeit, the adversary randomly selects e/c rows (for simplicity, assume that e is divisible by c), and compromises the results of sub-tasks corresponding to these rows, as illustrated in the middle plot of Fig. 5. Under this strategy, the

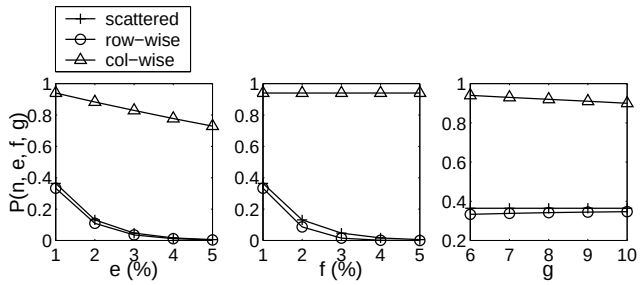

Fig. 6 Attack success rate of the adversary with different counterfeit strategies (under the column-based sampling scheme) with default setting: $n = 10000$, $e = 100$, $f = 100$, and $g = 6$.

probability that the adversary escapes from the f challenge objects embedded in the g columns is essentially equivalent to that all the challenge objects fall out of the overlapped region of the g sampled columns and the e/c compromised rows, formally

$$P_{row-wise}(e, f, g, n) = \frac{C^f_{g\cdot(r-\frac{e}{c})}}{C^f_{g\cdot r}}$$

Column-wise Counterfeit. Orthogonal to the row-wise counterfeit, within the column-wise scheme, the adversary chooses e/r columns, and compromises the sub-task results over all these columns, as illustrated in the rightmost plot of Fig. 5. Again, we are interested in calculating the probability that the adversary dodges the f challenge objects after compromising the results corresponding to these e sub-tasks. The calculation can be divided into two parts, the probability that i columns selected by the adversary overlap with that selected by the client, and the probability that no challenge objects appear in these i columns. Therefore,

$$P_{col-wise}(e, f, g, n) = \sum_{i=\max(0, g+\frac{e}{r}-c)}^{\min(g, \frac{e}{r})} \frac{C^i_{\frac{e}{r}} \cdot C^{g-i}_{c-\frac{e}{r}}}{C^g_c} \cdot \frac{C^f_{(g-i)\cdot r}}{C^f_{g\cdot r}} \quad (4)$$

where the first term corresponds to the probability that i columns selected by the client overlap with that selected by the adversary, and the second corresponds to that the f challenge objects fall out of the i intersected columns.

The attack success rate of the adversary with different counterfeit strategies with respect to varying setting of e, f and g is shown in Fig. 6. One can obtain the following interesting observations: 1) the success rate of the scattered strategy is independent of the parameter g, and decreases sharply as f or e grows; 2) the row-wise strategy offers the adversary with the least leverage for taking advantage of the correlations among the challenge objects; 3) the correlations of data blocks have the most significant impact on the col-wise strategy, with success rate close to one for all the settings! Clearly, this phenomenon stems from the unique behavior of our random sampling scheme: when the sampling rate (g/c) is low, the columns selected by the adversary and that by the client have extremely low chance to intersect.

3.3.4 A Revised Sampling Strategy

To remedy this drawback of the original random sampling scheme with respect to the column-wise counterfeit strategy, we introduce a hybrid sampling scheme: the client now issues cPIR on both column and row basises, i.e., the retrieved data blocks could be either column-wise and row-wise. Due to the space constraint, we omit the algorithmic details. Following, we analyze the verification power of this scheme, with respect to the above three counterfeit strategies.

Following the previous setting, except that instead of retrieving g columns, the client now retrieves g_c columns and g_r rows from the server, which form the pool for constructing the f challenge objects. Since the behavior of the scattered strategy is independent of the sampling scheme, we focus our discussion on the row-wise and column-wise strategies. Moreover, because of the symmetry, here we analyze the behavior of the column-wise strategy only. Now, the probability that the adversary dodges the f challenge objects after compromising the results corresponding to e/r columns of data blocks, can be formulated as follows: given that i columns selected by the server overlap with that selected by the client, it should be satisfied that the sampled sub-tasks selected by the client do not appear in 1) these i columns, or 2) the intersected region of the g_r rows (by the client) and the e/r columns (by the server). Therefore,

$$P^*_{col-wise}(e, f, g_r, g_c, n) = \sum_{i=\max(0, g_c+\frac{e}{r}-c)}^{\min(g_c, \frac{e}{r})} \frac{C^i_{\frac{e}{r}} \cdot C^{g_c-i}_{c-\frac{e}{r}}}{C^{g_c}_c} \cdot \frac{C^f_{(g_c-i)\cdot r+g_r\cdot(c+i-\frac{e}{r})-g_c\cdot g_r}}{C^f_{g_c\cdot r+g_r\cdot c-g_c\cdot g_r}}$$

Fig. 7 demonstrates the attack success rate of the adversary with different counterfeit strategies under our revised sampling scheme. It is clear that the hybrid sampling strategy is robust against both column-wise and row-wise counterfeits, with verification accuracy close to the ideal bound, but with much lower communication cost and computation burden on the server (f/g times of improvement). It is also interesting to notice that the enlargement of the sampled pool (g) does not improve the verification power, as shown in the rightmost plot of Fig. 7. What matters here is the ratio of the number of challenge objects over the size of the pool (f/g). This is good news, since the client can now achieve high verification accuracy with relative small g for fixed f, which implies low network communication cost.

Given the required verification accuracy λ, it is desirable to estimate the minimum values of g_c, g_r and f that achieve the least resource consumption. Formally, let ϕ_{row} and ϕ_{col} be the cost of retrieving a column and a row of data blocks respectively, and ψ_{con} be the construction cost of a challenge objects, one attempts to solve the following optimization problem:

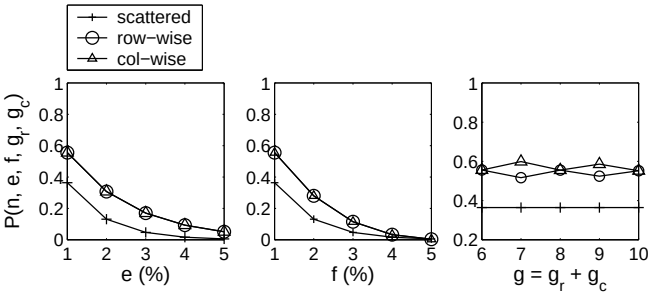

Fig. 7 Attack success rate of the adversary with different counterfeit strategies (under the hybrid sampling scheme), with default setting: n = 10000, e = 100, f = 100, and g = 6 ($g_c = \lceil g/2 \rceil$, $g_r = \lfloor g/2 \rfloor$).

$$\min \quad \phi_{row} \cdot g_r + \phi_{col} \cdot g_c + \psi_{con} \cdot f$$
$$\text{s.t.} \begin{cases} P^*_{row-wise}(n, e, f, g_r, g_c) \leq 1 - \lambda \\ P^*_{col-wise}(n, e, f, g_r, g_c) \leq 1 - \lambda \end{cases}$$

Unfortunately, no closed forms are available for optimal f, g_c and g_r; one can however apply off-the-shelf combinatorial optimization tools [29] to find approximate solutions.

4. Conclusion and Future Work

This paper presents a systematic study of the problem of providing execution assurance for massive computing tasks wherein both storage and computation are outsourced by untrusted third parties. We proposed a novel client-side checking solution by carefully intergrating the machineries of data authentication and computational private information retrieval (cPIR). This work also opens several directions for future research. First, how to effectively maintain the validity of the verification and challenge objects in facing frequent data updates? Second, how to incorporate more sophisticated optimization techniques, e.g., batch codes [30], to improve data retrieval efficiency? Third, how to construct more effective execution assurance solutions based on alternating primitives other than cPIR?

Acknowledgement

This work is partially sponsored by grants from NSF CISE NetSE program, CyberTrust program, and IBM faculty award, IBM SUR grant, and a grant from Intel research council.

References

[1] M. Kitsuregawa, “Challenge for info-plosion,” International conference on Algorithmic Learning Theory, 2007.
[2] “Amazon Elastic Compute Cloud (EC2): http://aws.amacon.com/ec2.”
[3] “Microsoft Azure Services Platform: http://www.microsoft.com/azure.”
[4] “Rackspace Mosso: http://www.mosso.com.”
[5] Y. Ohura, K. Takahashi, I. Pramudiono, and M. Kitsuregawa, “Experiments on query expansion for internet yellow page services using web log mining,” VLDB, 2002.
[6] P.T. Devanbu, M. Gertz, C.U. Martel, and S.G. Stubblebine, “Authentic third-party data publication,” IFIP Workshop on Database Security, 2001.
[7] F. Li, K. Yi, M. Hadjieleftheriou, and G. Kollios, “Proof-infused streams: enabling authentication of sliding window queries on streams,” VLDB, 2007.
[8] S. Papadopoulos, Y. Yang, and D. Papadias, “Cads: continuous authentication on data streams,” VLDB, 2007.
[9] G. Ateniese, R. Burns, R. Curtmola, J. Herring, L. Kissner, Z. Peterson, and D. Song, “Provable data possession at untrusted stores,” CCS, 2007.
[10] A. Juels and B.S. Kaliski, Jr., “Pors: proofs of retrievability for large files,” CCS, 2007.
[11] H. Shacham and B. Waters, “Compact proofs of retrievability,” ASIACRYPT, 2008.
[12] H. Pang and K.L. Tan, “Authenticating query results in edge computing,” ICDE, 2004.
[13] M. Narasimha and G. Tsudik, “Dsac: integrity for outsourced databases with signature aggregation and chaining,” CIKM, 2005.
[14] H. Pang, A. Jain, K. Ramamritham, and K.L. Tan, “Verifying completeness of relational query results in data publishing,” SIGMOD, 2005.
[15] F. Li, M. Hadjieleftheriou, G. Kollios, and L. Reyzin, “Dynamic authenticated index structures for outsourced databases,” SIGMOD, 2006.
[16] H. Pang and K. Mouratidis, “Authenticating the query results of text search engines,” Proc. VLDB Endow., vol.1, no.1, pp.126–137, 2008.
[17] W. Cheng and K.L. Tan, “Query assurance verification for outsourced multi-dimensional databases,” J. Comput. Secur., vol.17, no.1, pp.101–126, 2009.
[18] R. Sion, “Query execution assurance for outsourced databases,” VLDB, 2005.
[19] K. Yi, F. Li, M. Hadjieleftheriou, G. Kollios, and D. Srivastava, “Randomized synopses for query assurance on data streams,” ICDE, 2008.
[20] P. Golle and I. Mironov, “Uncheatable distributed computations,” CT-RSA, 2001.
[21] E. Kushilevitz and R. Ostrovsky, “Replication is not needed: single database, computationally-private information retrieval,” FOCS, 1997.
[22] R. Sion, “On the computational practicality of private information retrieval,” NDSS, 2007.
[23] G.S. Graham and P.J. Denning, “Protection: principles and practice,” AFIPS spring joint computer conference, 1972.
[24] S. Osborn, R. Sandhu, and Q. Munawer, “Configuring role-based access control to enforce mandatory and discretionary access control policies,” ACM Trans. Inf. Syst. Secur., vol.3, no.2, pp.85–106, 2000.
[25] National Institute of Standards and Technologies, “FIPS PUB 180-1: secure hash standard,” 1995.
[26] S. Goldwasser, S. Micali, and R.L. Rivest, “A digital signature scheme secure against adaptive chosen-message attacks,” SIAM J. Comput., vol.17, no.2, pp.281–308, 1988.
[27] C. Cachin, S. Micali, and M. Stadler, “Computationally private information retrieval with polylogarithmic communication,” EUROCRYPT, 1999.
[28] R.C. Merkle, “A certified digital signature,” CRYPTO, 1989.
[29] C.H. Papadimitriou and K. Steiglitz, Combinatorial optimization: algorithms and complexity, Prentice-Hall, Inc., 1982.
[30] Y. Ishai, E. Kushilevitz, R. Ostrovsky, and A. Sahai, “Batch codes and their applications,” STOC, 2004.