A Practical Approach to the Attestation of Computational Integrity in Hybrid Cloud

Andrew Pawloski, Longfei Wu, Xiaojiang Du
Dept. of Computer and Information Science
Temple University
Philadelphia, Pennsylvania 19122
{apawloski, longfei.wu, dux}@temple.edu

Lijun Qian
Dept. of of Electrical and Computer Engineering
Prairie View A&M University
Prairie View, Texas 77446
liqian@pvamu.edu

***Abstract*—In the context of a heterogeneous trust environment, such as a hybrid cloud (consisting of public and private cloud), it is possible for an untrusted component to create a Byzantine fault intentionally. For example, a public cloud service provider may only do a partial (instead of a complete) database search just to save its computation. These malicious events can be particularly dangerous because the system does not crash but an incorrect and actionable result will be produced. Although there are ways to provide malicious Byzantine Fault-Tolerance in classical distributed systems, their computation and communication properties make them infeasible to be used in the hybrid cloud environment. In this paper, we present an efficient method for attestation of computational integrity in hybrid cloud.**

Keywords*—*Hybrid cloud; computational integrity; attestation

I. Introduction

A. The Hybrid Cloud

The rise of ”cloud” computing over the past seven years has led to a revolution in organizational computation. What once required massive internal resources (e.g. large computing clusters, storage centers, and the requisite maintenance associated with each) can now be outsourced to third parties, like Amazon, IBM, or Microsoft, for a fraction of the cost. This reduction of operational overhead has allowed organizations to approach ”big-data” problems in ways which were previously infeasible (for example, the New York Times – an organization with little need for a permanent internal cluster – processed 4TB of image data into 11 million digitized articles by renting less than 24 hours of time on Amazon’s Elastic Compute Cloud (EC2) [1]).

Despite this attractiveness, however, the cloud computing revolution is at the fundamental level a case of outsourcing. Thus, an organization must trust its cloud provider to not be malicious or incompetent with any information entrusted to it. For clients with sensitive data – for example, secret internal documents or private customer information – trusting a third party can entail a significant level of risk [2] [3].

One method of mitigating this risk that has seen recent popularity is ”hybrid” cloud computing, wherein an organization uses two clouds – a ”private” internal cloud and a ”public” third party cloud – to fulfill computational and storage needs [4]. In the typical usage scenario, an organization runs computations on one of the clouds depending on the sensitivity level of the data. (More specifically, sensitive data is sequestered on the trusted private cloud, while non-sensitive data can be outsourced to the public cloud.) This allows a client to take advantage of the low cost and scalability of a public cloud without the risk of leaking or losing important private data.

Although the hybrid cloud allows for greater privacy, it brings with it some unique properties which must be accommodated (and which preclude many public-cloud-only applications from being effective on the hybrid cloud) [5]. A quick summary of these properties can be seen in Fig. 1. The most important of these properties is communication speed: instead of communicating within the same server rack (or at least the same data center), hybrid clouds are likely to be geographically distant. Thus communication speed becomes the most significant bottleneck, and applications which are communication-bound can become virtually unusable on the hybrid cloud.

Currently, most research focuses on utilizing hybrid clouds to maintain privacy during important computations [6], [7]. However, an equally important area in this emerging field is the attestation of computational integrity. We attempt to address this through a robust auditing framework specifically designed for hybrid clouds that ensures components return honest results.

B. Malicious Byzantine Faults

Our framework’s specific targets are essentially malicious Byzantine faults. A Byzantine fault occurs when the misbehavior of one or more components in a distributed system goes undetected (that is, the system doesn’t halt or crash because of the misbehavior) [8]. We’re interested in a special subset of these, in which an adversary intentionally returns malicious results to the private cloud, thereby giving the client actionable but incorrect information. There are a variety of situations in which this might occur: one scenario would be a cloud provider who has had a data loss event but refuses to admit it to the client (returning ”spoofed” results from the public cloud instead); another scenario would be an adversary who wants prevent the client from discovering something in the public cloud data (returning ”scrubbed” results from the public cloud instead).

It should be noted that although we classify our targets as Byzantine faults, we use the term loosely. Our framework is not a true Byzantine fault tolerance (BFT) system, but

978-1-4799-6959-3/15/$31.00 ©2015 IEEE

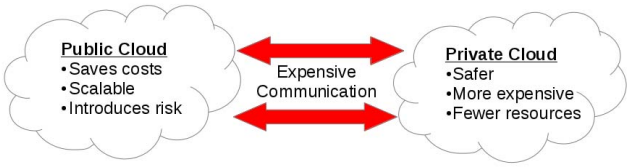

Fig. 1. Hybrid Cloud Considerations

rather a practical, efficient, and generalized method to perform computational integrity attestation across heterogeneous trust environments. We make this consideration by noting that organizations which require true BFT – those who could face catastrophic failure should a Byzantine fault occur – are unlikely to entrust a third party cloud for their computations in the first place.

The rest of the paper is organized as follows. Section II provides an overview, including the system model, threat model, detection efficacy, and the theoretical overhead. Section III presents our framework for detecting the malicious Byzantine faults. Section IV is the performance evaluation. Section V discusses related works, and Section VI concludes the paper.

II. Overview

A. The System Model

Our framework is optimized and directly intended for use on hybrid clouds. While there are currently effective methods of attestation in classical distributed systems, our intention is to address this problem in consideration of the emerging hybrid cloud architecture. We define the hybrid cloud to be a system containing the following components:

1) The Public Cloud: The public cloud – generally referred to as just ”the cloud” in most other contexts – is any third party entrusted with information for either computation or storage. Examples of public cloud providers are Amazon, IBM, NTT Communications, and Virtustream [9]. Although these are respected providers, because their resources are not controlled by the client, we deem them to be untrusted.

2) The Private Cloud: We define the private cloud to be any internal computational or storage resource. More generally, we consider this to include any resource which is entirely controlled by the client. We make the important assumption that the private cloud is competently maintained (especially in terms of avoiding system failures and security breaches).

There are two scenarios where a hybrid cloud is particularly useful. The first is short-term infrastructure scaling. In this case, an organization expects a temporarily excessive load on their internal system and uses a public resource to offset the additional work [10]. The second scenario, as described by Zhang et al [7], occurs when an organization has to store or work across a dataset with both sensitive and non-sensitive information. In this case, the sensitive information is maintained solely on the private cloud, while the non-sensitive information is outsourced. We are interested in this latter scenario.

We also make an assumption about the relative sizes of the public and private clouds. Because a major advantage of the public cloud is low cost, we assume that it is reasonable for a public cloud to be larger than or equal to the private cloud in size.

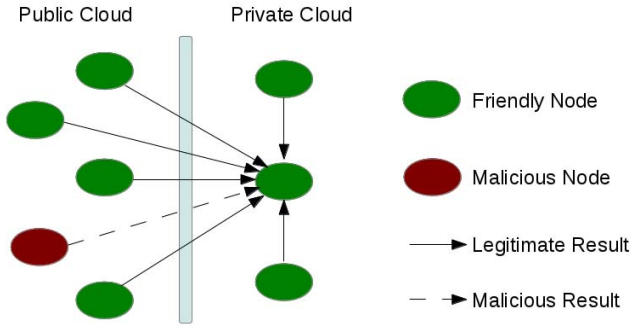

Fig. 2. Threat Model

Finally, our model focuses on the computational functionalities of hybrid clouds (that is, we do not address the scenario where a hybrid cloud is used for storage). More specifically, our model assumes that all reductions over both sensitive and non-sensitive data occur on the private cloud. This is the only way to prevent the public cloud from accessing private information.

B. The Threat Model

Our framework is based on the assumption that, at any given time, a node in the private cloud can be trusted. Thus our threat model is restricted to the public cloud, which has the ability to ”cheat” and cause a malicious Byzantine fault. We assume that untrusted nodes will not return results that will halt the distributed computation but rather will return results that throw off the final output delivered to the client. The frequency of these malicious faults can vary between 0 and 100% of the results returned by the public cloud. We also assume that all computational reductions must occur on the private cloud in order to preserve the confidentiality of private cloud data. Finally, we assume that the public cloud nodes cheat at random – or at least cheat in a way which cannot be predicted by the client. This model is visualized in Fig. 2.

C. Detection Efficacy

Our framework works through strategic redundancy. A sample of untrusted nodes is selected at random and their work is duplicated on the private cloud. The sample size depends on the expected ”cheat” rate and desired confidence level. The relationship is defined by the following function:

$$P(k) = 1 - (1 - p)^k \quad (1)$$

where n is the total number of nodes, k is the number of nodes sampled, p is the probability of a public cloud node cheating, and $P(k)$ is the probability of catching any such cheating in our sample. Fig. 3 indicates the detection rate versus number of nodes per sample and can be used to determine sample size. Depending on the cheat rate, a reasonable sample size for a computation of X nodes can be anywhere between Y and Z nodes. Because we assume the private cloud is trusted, the attestation itself is simply a comparison of the results of the redundant computations.

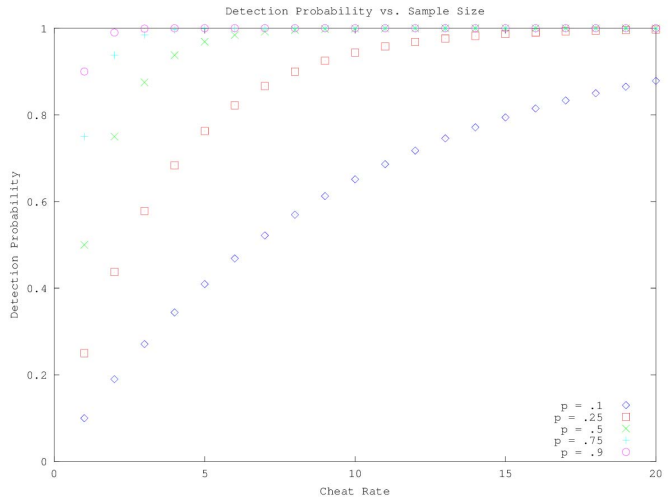

Fig. 3. Sample detection rate graph

D. Theoretical Overhead

The most important consideration for our framework is minimizing intercloud communication. It is because of this bottleneck that otherwise effective systems [11] [12] – which require frequent communication between nodes – are less than optimal in hybrid cloud environments. For a clique-based algorithm in a bipartite system (like a hybrid cloud) with m public nodes and n private nodes, there is a communicative complexity of $O(mn)$. Because communication is the most expensive part of hybrid cloud computing, it is worthwhile to decrease this complexity.

Because we can assume that final computational reductions occur in the private cloud (otherwise the untrusted cloud would be given private data), we can take advantage of the fact that untrusted results are going to be communicated anyway. Thus, in many cases, our framework can operate successfully with virtually no additional intercloud communication. In the worst case scenario, a single file is moved from the public cloud to the private cloud – as will be discussed in section three, this introduces an overhead of 160 bits per public node.

III. Our Framework

To detect and avoid these malicious Byzantine faults, we developed a modularized runtime framework to abstract any attestation away from the end user. This is done with three modules: ”public_compute,” ”private_compute” and ”validate.” The latter is predefined, while the former two are written by the user.

The user-defined modules contain the instructions for the distributed application itself. The nature and structure of the compute modules can be decided by the user with one exception: the compute modules must take an integer rank as a parameter. (Arguments are passed to the compute modules by our environment.) We will discuss momentarily why this is necessary. The only other requirement of the user is to designate the data required by the public cloud. By default, all files are considered private unless otherwise classified by the user. The public data is migrated to the public cloud, but – because our private cloud will be running a sample of public computations – a copy is retained internally.

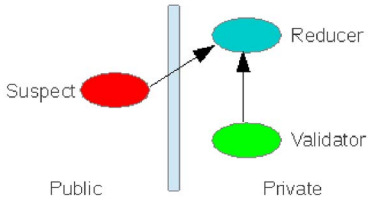

Fig. 4. Attestation when there is no Public-side reduction

At runtime, our attestation environment is called and passed any arguments needed by the compute modules. The environment detects the nodes in each cloud and assigns each a unique designation (an integer ID). Then, one of two sequences of events occurs:

A. Public (Untrusted) Cloud

In the public cloud, each node runs a separate instance of the public_compute module, passing its integer rank as an argument. The nodes operate normally. When a node’s computation is finished and ready to be returned to the private cloud, the result is hashed, signed, and saved to a log file resident on the untrusted cloud.

B. Private (Trusted) Cloud

Like its public counterpart, the private cloud also runs its respective module (private_compute) on each node, passing the node’s integer rank as an argument. However in addition to the normal computation, a series of other steps are performed.

First, the master node (arbitrarily chosen to be the private node of lowest rank) randomly selects the sample of public nodes whose results are to be validated. Then, the master node randomly decides a private node (or nodes, depending on workload) to perform the redundant computations. We refer to these nodes as ”validator nodes.” The redundant computations are trivial to run; a private cloud node simply calls the public_compute function and passes it the integer ID of the node that’s to be validated. This can be done when the system is in an asynchronous state.

Then, depending on the nature of the distributed computation, one of two situations will occur:

1) A Public-side Reduction Has Not Occurred: Once a private node finishes the computation, it sends the result to the respective reduction node in the same cloud. This is the node which is to receive the challenged public node’s result organically (as a part of the actual hybrid distributed computation) and is decided through rank, or can be manually overridden by the user. The reduction node then compares the results received from the challenged public node with the results from the trusted redundant computation. This is illustrated in Fig. 4. If there is an inconsistency between the two, then recovery is trivial: the reduction node simply uses the result submitted by the trusted node instead.

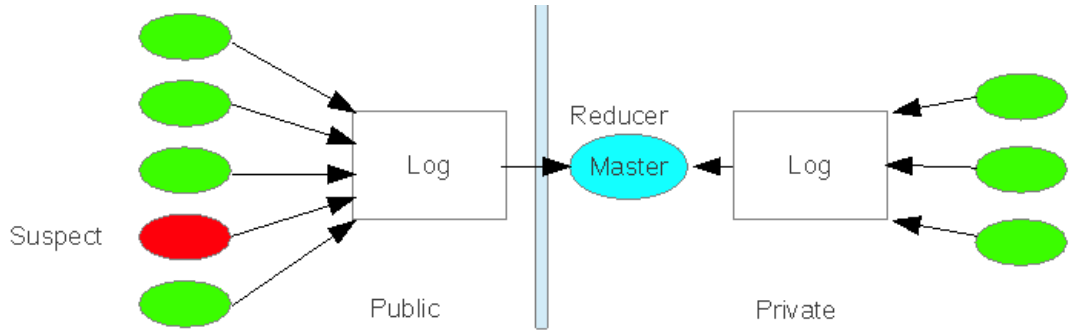

Fig. 5. Attestation when a Public-side reduction has occurred

2) A Public-side Reduction Has Occurred: If a reduction has already occurred on the untrusted cloud (for example, multiple word counts across many documents were consolidated into a single overall count) and consolidated results were returned to the private cloud, then validation is a little more involved. In this case, the log file with the hashed results from each untrusted node is transmitted from the public cloud to the master node of the private cloud. This incurs an intercloud communication overhead of $20*n$ bytes, where n is the number of nodes in the public cloud.

In this situation, instead of passing them to the respective reducers, the private validator nodes hash their results and store them into a log file in a manner identical to the public nodes. Once this process is complete, the master node then compares the log files. This is illustrated in Fig. 5. If there is an inconsistency between two entries, then recovery is a little more involved. Because a reduction has already occurred, a malicious result cannot simply be replaced with a trusted one. Instead there are multiple options: 1) because the log entries are signed, we can quickly identify a malicious node. Therefore the client may want to remove the offender from eligible public nodes and repeat the process. 2) Because the untrusted cloud now has confirmed-malicious nodes, the client can offset the computations back on the private cloud. This is likely impractical due to the fact that a hybrid cloud was necessary in the first place. 3) The client can simply halt the distributed computation and continue later with a different public cloud provider.

IV. Evaluation

We first tested our framework on an internal server cluster, running the simulation 100 times on 1 through 8 servers. We timed the computation and validation sections of the program and plotted the average CPU share for each. The results can be seen in Fig. 6. More specifically, the typical validation share ranged from 6.86% (1 server) to 1.04% (8 servers).

On the same internal cluster, we also implemented a large textual analysis computation across a large collection of English documents. As in the first experiment, we timed the computation and validation section. A plot of the overheads associated with each can be seen in Fig. 7.

V. Related Work

Hybrid cloud is an emerging model of cloud deployment that involves both private cloud within an organization and public commercial cloud. This nature allows organizations to keep sensitive data in trusted private cloud while outsource insensitive data to public cloud. Prometheus [13] is a privacy-

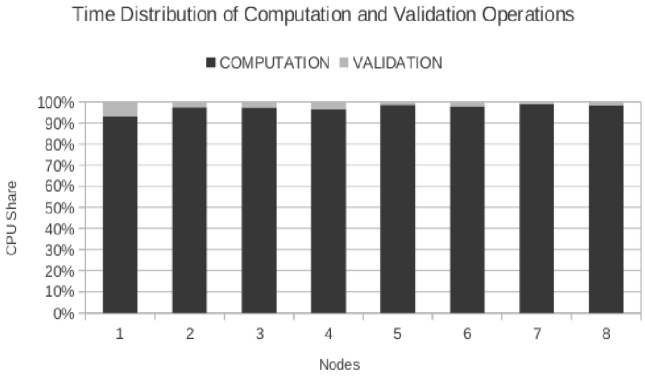

Fig. 6. Runtime Overhead

TABLE I. Percent CPU Time for Select Tests

Servers	1	5	6	8
Computation	93.14	97.99	98.61	98.95
Validation	6.86	2.00	1.38	1.04

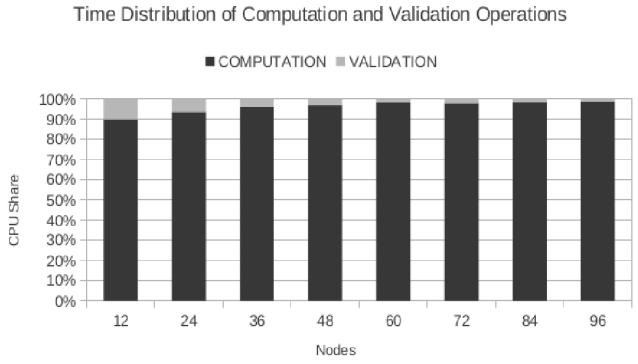

Fig. 7. Runtime Overhead for Test Two

TABLE II. Percent CPU Time for Select Tests

Servers	1	5	6	7	8
Computation	90.12	97.96	98.20	98.49	98.96
Validation	9.88	2.04	1.80	1.51	1.34

aware data retrieval system that can automatically separate sensitive information from public data. Zhang et. al proposed Sedic [7], a secure data-intensive computing system which can not only automatically partition a computing job according to the security levels of the data it works on, but also arrange the computation across hybrid cloud. Specifically, they built an analysis tool that automatically evaluates and transforms the reduction structure of a computing job to optimize it for hybrid-cloud computing. Zhu et. al [14] presented a collaborative integrity verification mechanism in hybrid clouds based on the techniques of homomorphic verifiable responses and hash index hierarchy. But these works focus on either leaking of private data or data integrity and dynamic scalability in hybrid cloud storage. We consider the important issue of computational integrity given the public cloud services cannot be (fully) trusted.

Several existing schemes have been proposed to solve the integrity assurance vulnerability of computation result, especially for the most popular cloud computing paradigm – MapReduce. Wei et al. proposed SecureMR [15], a practical service integrity assurance framework for MapReduce. Specifically, they implemented a scalable decentralized replication-based verification scheme to protect the integrity of MapReduce data processing service. HybrEx [16] checks the integrity of the results from the public cloud in two modes that provide different levels of fidelity. The first mode is full integrity checking, where the private cloud re-executes every Map and Reduce task that the public cloud has executed. Another one is quick integrity checking, where the private cloud selectively checks the integrity of the results from the public cloud. Wang et al. [17] proposed the Verification-based Integrity Assurance Framework (VIAF) to detect both non-collusive and collusive mappers. The basic idea of VIAF is to combine task replication with non-deterministic verification, in which consistent but malicious results from collusive mappers can be detected. Our sampling based scheme performs the verification on private cloud, which is also able to detect collusive workers. However, we focus on the computational integrity in hybrid cloud.

VI. Conclusion

The increasing organizational reliance on cloud computing has reintroduced a wave of concerns about trusting third parties with sensitive information. Although hybrid cloud computing has been introduced as a practical way to mitigate these aforementioned risks, most of the current research focuses primarily on remaining anonymous in such a context. In this paper, we introduced a framework which allows for lightweight computation attestation specifically designed with the logistical considerations of hybrid clouds in mind. By taking advantage of some natural properties of distributed computing in a hybrid cloud (particularly private-side reduction) we enable this attestation to occur with virtually no (or at most a trivial amount of) intercloud overhead. Finally, we evaluated the system on our internal servers and found the performance to be adequate for most computational contexts.

Acknowledgement

This work was supported in part by the US National Science Foundation under grants CNS-0963578, CNS-1022552, and CNS-1065444.

References

[1] M. Zaharia, A. Konwinski, A. D. Joseph, R. Katz, and I. Stoica, "Improving mapreduce performance in heterogeneous environments," in *Proceedings of the 8th USENIX conference on Operating systems design and implementation*, ser. OSDI'08. Berkeley, CA, USA: USENIX Association, 2008, pp. 29–42.

[2] M. Jensen, J. Schwenk, N. Gruschka, and L. Iacono, "On technical security issues in cloud computing," in *Cloud Computing, 2009. CLOUD '09. IEEE International Conference on*, 2009, pp. 109–116.

[3] B. Kandukuri, V. Paturi, and A. Rakshit, "Cloud security issues," in *Services Computing, 2009. SCC '09. IEEE International Conference on*, 2009, pp. 517–520.

[4] P. Mell and T. Grance, "The nist definition of cloud computing," *Information Technology Laboratory (National Institute of Standards and Technology)*, 2011.

[5] Q. Zhang, L. Cheng, and R. Boutaba, "Cloud computing:state-of-the-art and research challenges," *Journal of Internet Services and Applications*, May 2010.

[6] K.-M. Chung, Y. Kalai, and S. Vadhan, "Improved delegation of computation using fully homomorphic encryption," in *Proceedings of the 30th annual conference on Advances in cryptology*, ser. CRYPTO'10. Berlin, Heidelberg: Springer-Verlag, 2010, pp. 483–501.

[7] K. Zhang, X. Zhou, Y. Chen, X. Wang, and Y. Ruan, "Sedic: privacy-aware data intensive computing on hybrid clouds," in *Proceedings of the 18th ACM conference on Computer and communications security*, ser. CCS '11. New York, NY, USA: ACM, 2011, pp. 515–526.

[8] M. Castro, "Practical byzantine fault tolerance," Thesis (Ph. D.)–Massachusetts Institute of Technology, Dept. of Electrical Engineering and Computer Science, February 2001.

[9] A. Li, X. Yang, S. Kandula, and M. Zhang, "Cloudcmp: comparing public cloud providers," in *Proceedings of the 10th ACM SIGCOMM conference on Internet measurement*, ser. IMC '10. New York, NY, USA: ACM, 2010, pp. 1–14. [Online]. Available: http://doi.acm.org/10.1145/1879141.1879143

[10] M. Armbrust, A. Fox, R. Griffith, A. D. Joseph, R. Katz, A. Konwinski, G. Lee, D. Patterson, A. Rabkin, I. Stoica, and M. Zaharia, "A view of cloud computing," *Commun. ACM*, vol. 53, no. 4, pp. 50–58, Apr. 2010.

[11] J. Du, N. Shah, and X. Gu, "Adaptive data-driven service integrity attestation for multi-tenant cloud systems," in *Quality of Service (IWQoS), 2011 IEEE 19th International Workshop on*, 2011, pp. 1–9.

[12] J. Du, W. Wei, X. Gu, and T. Yu, "Runtest: assuring integrity of dataflow processing in cloud computing infrastructures," in *Proceedings of the 5th ACM Symposium on Information, Computer and Communications Security*, ser. ASIACCS '10. New York, NY, USA: ACM, 2010, pp. 293–304.

[13] Z. Zhou, H. Zhang, X. Du, P. Li, and X. Yu, "Prometheus: Privacy-aware data retrieval on hybrid cloud," in *INFOCOM, 2013 Proceedings IEEE*, 2013, pp. 2643–2651.

[14] Y. Zhu, H. Hu, G.-J. Ahn, Y. Han, and S. Chen, "Collaborative integrity verification in hybrid clouds," in *Collaborative Computing: Networking, Applications and Worksharing (CollaborateCom), 2011 7th International Conference on*, 2011, pp. 191–200.

[15] W. Wei, J. Du, T. Yu, and X. Gu, "Securemr: A service integrity assurance framework for mapreduce," in *Computer Security Applications Conference, 2009. ACSAC '09. Annual*, 2009, pp. 73–82.

[16] S. Y. Ko, K. Jeon, and R. Morales, "The hybrex model for confidentiality and privacy in cloud computing," in *Proceedings of the 3rd USENIX conference on Hot topics in cloud computing*, ser. HotCloud'11. Berkeley, CA, USA: USENIX Association, 2011, pp. 8–8.

[17] Y. Wang and J. Wei, "Viaf: Verification-based integrity assurance framework for mapreduce," in *Cloud Computing (CLOUD), 2011 IEEE International Conference on*, 2011, pp. 300–307.