# Integrity and Consistency for Untrusted Services (Extended Abstract)

Christian Cachin

IBM Research - Zurich
CH-8803 Rüschlikon, Switzerland
cca@zurich.ibm.com

21 January 2011

**Abstract**

A group of mutually trusting clients outsources an arbitrary computation service to a remote provider, which they do not fully trust and that may be subject to attacks. The clients do not communicate with each other and would like to verify the integrity of the stored data, the correctness of the remote computation process, and the consistency of the provider's responses.

We present a novel protocol that guarantees atomic operations to all clients when the provider is correct and fork-linearizable semantics when it is faulty; this means that all clients which observe each other's operations are consistent, in the sense that their own operations, plus those operations whose effects they see, have occurred atomically in same sequence. This protocol generalizes previous approaches that provided such guarantees only for outsourced storage services.

## 1 Overview

Today many users outsource generic computing services to large-scale remote service providers and no longer run them locally. Commonly called the *cloud computing model*, this approach carries inherent risks concerning data security and service integrity.

Whereas data can be stored confidentially by encrypting it, ensuring the *integrity* of remote data and outsourced computations is a much harder problem. A subtle change in the remote computation, whether caused inadvertently by a bug or deliberately by a malicious adversary, may result in wrong responses to the clients. Such deviations from a correct specification can be very difficult to spot manually.

Suppose a group of clients, whose members trust each other, relies on an untrusted remote server for a collaboration task. For instance, the group stores its project data on a cloud service and accesses it for coordination and document exchange. Although the server is usually correct and responds properly, it might become corrupted some day and respond wrongly. This work aims at discovering such misbehavior, in order for the clients to take some compensation action.

When the service provides data storage (read and write operations only), some well-known methods guarantee data integrity. With only one client, a *memory checker* [1] ensures that a read operation always returns the most recently written value. If multiple clients access the remote storage, they can combine a memory checker with an external trusted infrastructure (like a directory service or a key manager in a cryptographic file system), and achieve the same guarantees for many clients.

But in the asynchronous network model without client-to-client communication considered here, nothing prevents the server from mounting a *forking attack*, whereby it simply omits the operations of

one client in its responses to other clients. Mazières and Shasha [14] put forward the notion of *fork-linearizability*, which captures the optimal achievable consistency guarantee in this setting. It ensures that whenever the server's responses to a client $A$ have ignored a write operation executed by a client $B$, then $A$ can never again read a value written by $B$ afterwards and vice versa. With this notion, the clients detect server misbehavior from a single inconsistent operation — this is much easier than comparing the effects of *all* past operations one-by-one.

This paper makes the first step toward ensuring integrity and consistency for *arbitrary computing services* running on an untrusted server. It does so by extending untrusted storage protocols providing fork-linearizability to a generic service protocol with fork-linearizable semantics. Previous work in this model only addressed integrity for a storage service, but could not check the consistency of more general computations by the server.

Similar to the case of a storage service, the server can readily mount a forking attack by splitting the group of clients into subgroups and responding consistently within each subgroup, but not making operations from one subgroup visible to others. Because the protocol presented here ensures fork-linearizability, however, such violations become easy to discover. The method therefore protects the integrity of arbitrary services in an end-to-end way, as opposed to existing techniques that aim at ensuring the integrity of a computing platform (e.g., the *trusted computing* paradigm).

Our approach requires that (at least part of) the service implementation is known to the clients, because they need to double-check crucial steps of an algorithm locally. In this sense, the notion of fork-linearizable service integrity, as considered here, means that the clients have collaboratively verified every single operation of the service. This strictly generalizes the established notion of fork-linearizable storage integrity. A related notion for databases is ensured by the Blind Stone Tablet protocol [18].

## 2 Contributions

We present the first precise model for a group of mutually trusting clients to execute an *arbitrary service* on an untrusted server $S$, with the following characteristics. It guarantees *atomic operations* to all clients when $S$ is correct and *fork-linearizability* when $S$ is faulty; this means that all clients which observe each other's operations are *consistent*, in the sense that their own operations, plus those operations whose effects they see, have occurred atomically in same sequence.

Furthermore, we generalize the concept of *authenticated data structures* [15] toward executing arbitrary services in an authenticated manner with multiple clients. We present a protocol for consistent service execution on an untrusted server, which adds $O(n)$ communication overhead for a group of $n$ clients; it generalizes existing protocols that have addressed only the special case of storage on an untrusted server.

## 3 Related work

Ensuring integrity and consistency for services outsourced to third parties is a very important problem, particularly regarding security in cloud computing [8].

A common approach for tolerating faults, including adversarial actions by malicious, so-called Byzantine servers, relies on replication [5]. All such methods, however, break down as soon as a majority of servers becomes faulty. We are interested in consistency for only one server, which is potentially Byzantine.

Our approach directly builds on *authenticated data structures* [15, 13, 17]; they generalize Merkle hash trees for memory checking [1] to arbitrary search structures on general data sets. Authenticated data structures consist of communication-efficient methods for authenticating database queries answered

by an untrusted provider. In contrast to our setting, the two- and three-party models of authenticated data structures allow only one client as a writer to modify the content. Our model allows any client to issue arbitrary operations, including updates.

Previous work on untrusted storage has addressed the multi-writer model. Mazières and Shasha [14] introduce untrusted storage protocols and the notion of fork-linearizability (under the name of *fork consistency*), and demonstrate them with the SUNDR storage system [11]. Subsequent work of Cachin et al. [4] improves the efficiency of untrusted storage protocols. A related work demonstrates how the operations of a revision control system can be mapped to an untrusted storage primitive, such that the resulting system protects integrity and consistency for revision control [2].

FAUST [3] and Venus [16] extend the model beyond the one considered here and let the clients occasionally exchange messages among themselves. This allows FAUST and Venus to obtain stronger semantics, in the sense that they eventually reach consistency (in the sense of linearizability) or detect server misbehavior. In our model without client-to-client communication, fork-linearizability, or one of the related "forking" consistency notions [3], is the best that can be achieved [14].

Several recent cloud-security mechanisms aim at a similar level of service consistency as guaranteed by our protocol. They include the Blind Stone Tablet [18] for consistent and private database execution using untrusted servers, the SPORC framework [9] for securing group collaboration tasks executed by untrusted servers, and the Depot [12] storage system.

Orthogonal approaches impose correct behavior on a remote service indirectly, for instance through accountability in a storage service [19] or distributed systems [10]. Yet other work relies on trusted hardware modules at all parties [6, 7].

## Acknowledgments

This work has been supported in part by the European Commission through the ICT programme under contracts ICT-2007-216676 ECRYPT II and ICT-2009-257243 TCLOUDS.

## References

[1] M. Blum, W. Evans, P. Gemmell, S. Kannan, and M. Naor, "Checking the correctness of memories," *Algorithmica*, vol. 12, pp. 225–244, 1994.

[2] C. Cachin and M. Geisler, "Integrity protection for revision control," in *Proc. Applied Cryptography and Network Security (ACNS)* (M. Abdalla and D. Pointcheval, eds.), vol. 5536 of *Lecture Notes in Computer Science*, pp. 382–399, Springer, 2009.

[3] C. Cachin, I. Keidar, and A. Shraer, "Fail-aware untrusted storage," in *Proc. International Conference on Dependable Systems and Networks (DSN-DCCS)*, pp. 494–503, 2009.

[4] C. Cachin, A. Shelat, and A. Shraer, "Efficient fork-linearizable access to untrusted shared memory," in *Proc. 26th ACM Symposium on Principles of Distributed Computing (PODC)*, pp. 129–138, 2007.

[5] B. Charron-Bost, F. Pedone, and A. Schiper, eds., *Replication: Theory and Practice*, vol. 5959 of *Lecture Notes in Computer Science*. Springer, 2010.

[6] B.-G. Chun, P. Maniatis, S. Shenker, and J. Kubiatowicz, "Attested append-only memory: Making adversaries stick to their word," in *Proc. 21st ACM Symposium on Operating System Principles (SOSP)*, pp. 189–204, 2007.

[7] B.-G. Chun, P. Maniatis, S. Shenker, and J. Kubiatowicz, "Tiered fault tolerance for long-term integrity," in *Proc. 7th USENIX Conference on File and Storage Technologies (FAST)*, 2009.

[8] Cloud Security Alliance (CSA). `http://www.cloudsecurityalliance.org/`, 2010.

[9] A. J. Feldman, W. P. Zeller, M. J. Freedman, and E. W. Felten, "SPORC: Group collaboration using untrusted cloud resources," in *Proc. 9th Symp. Operating Systems Design and Implementation (OSDI)*, 2010.

[10] A. Haeberlen, P. Kouznetsov, and P. Druschel, "PeerReview: Practical accountability for distributed systems," in *Proc. 21st ACM Symposium on Operating System Principles (SOSP)*, pp. 175–188, 2007.

[11] J. Li, M. Krohn, D. Mazires, and D. Shasha, "Secure untrusted data repository (SUNDR)," in *Proc. 6th Symp. Operating Systems Design and Implementation (OSDI)*, pp. 121–136, 2004.

[12] P. Mahajan, S. Setty, S. Lee, A. Clement, L. Alvisi, M. Dahlin, and M. Walfish, "Depot: Cloud storage with minimal trust," in *Proc. 9th Symp. Operating Systems Design and Implementation (OSDI)*, 2010.

[13] C. Martel, G. Nuckolls, P. Devanbu, M. Gertz, A. Kwong, and S. G. Stubblebine, "A general model for authenticated data structures," *Algorithmica*, vol. 39, pp. 21–41, 2004.

[14] D. Mazières and D. Shasha, "Building secure file systems out of Byzantine storage," in *Proc. 21st ACM Symposium on Principles of Distributed Computing (PODC)*, 2002.

[15] M. Naor and K. Nissim, "Certificate revocation and certificate update," *IEEE Journal on Selected Areas in Communications*, vol. 18, pp. 561–570, Apr. 2000.

[16] A. Shraer, C. Cachin, A. Cidon, I. Keidar, Y. Michalevsky, and D. Shaket, "Venus: Verification for untrusted cloud storage," in *Proc. Cloud Computing Security Workshop (CCSW)*, ACM, 2010.

[17] R. Tamassia and N. Triandopoulos, "Computational bounds on hierarchical data processing with applications to information security," in *Proc. 32nd International Colloquium on Automata, Languages and Programming (ICALP)* (L. Caires *et al.*, eds.), vol. 3580 of *Lecture Notes in Computer Science*, pp. 153–165, Springer, 2005.

[18] P. Williams, R. Sion, and D. Shasha, "The blind stone tablet: Outsourcing durability to untrusted parties," in *Proc. Network and Distributed Systems Security Symposium (NDSS)*, 2009.

[19] A. R. Yumerefendi and J. S. Chase, "Strong accountability for network storage," *ACM Transactions on Storage*, vol. 3, no. 3, 2007.