Integrity and Consistency for Untrusted Services
(Extended Abstract)

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

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.

Keywords: cloud computing, fork-linearizability, data integrity, computation integrity, authenticated data structure, Byzantine emulation.

1 Introduction

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 \[15\] 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 \[20\].

### 1.1 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* \[16\] 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.