Chapter 8
Implementing Trustworthy Services Using Replicated State Machines

Fred B. Schneider and Lidong Zhou

Abstract A thread of research has emerged to investigate the interactions of replication with threshold cryptography for use in environments that satisfy weak assumptions. The result is a new paradigm known as distributed trust, and this chapter attempts to survey that landscape.

8.1 Introduction

“Divide and conquer” can be a powerful tool for disentangling complexity when designing a computing system. However, some aspects of a system design are inseparable. Treating these as though they were independent leads to one interfering with the other, and “divide and be conquered” perhaps better characterizes the consequences. For some years, we have been investigating how to construct systems that continue functioning despite component failures and despite attacks. A question we have pondered is to what extent does divide and conquer apply? Somewhat less than you might hope is, unfortunately, the answer.

One could argue that attacks can be seen as just another cause for component failure. The Byzantine fault model asserts that a faulty component can exhibit arbitrarily malicious (so-called “Byzantine”) behavior; a system that tolerates Byzantine faults should then be able to handle anything. Moreover, because any component can be viewed abstractly in terms of its state and a set of possible next-state transitions—in short, a state machine—fault-tolerant services could be built by assembling enough state machine copies so that outputs from the ones exhibiting Byzantine behavior are outvoted by the correctly functioning ones. The fault-tolerance of the ensemble thus exceeds the fault-tolerance of any individual state machine, and a distributed fault-tolerance is the result.

A closer look at such replicated state machines, however, reveals problems when attacks are possible. Specific difficulties with the approach and how we can over-
come these are described later in this chapter, but the overall vision remains compelling: placing more trust in an ensemble than in any of its individual components. In analogy with distributed fault-tolerance, then, we are seeking ways to implement distributed trust.

8.2 The State-Machine Approach

The details for using replicated state machines and implementing a Byzantine fault-tolerant service [19, 29] are well-known.

1. Start with a server, structured as a deterministic state machine, that reads and processes client submitted requests, which are the sole means to change the server’s state or cause the server to produce an output.
2. Run replicas of that server on distinct hosts. These hosts communicate through narrow-bandwidth channels and thus form a distributed system.
3. Employ a replica-coordination protocol to ensure that all non-faulty server replicas process identical sequences of requests.

Correctly operating server replicas will produce identical outputs for each given client request. Moreover, the majority of the outputs produced for each request will come from correct replicas provided that at most \( t \) server replicas are faulty and that the service comprises at least \( 2t + 1 \) server replicas. So, we succeed in implementing availability and integrity for a service that tolerates at most \( t \) faulty replicas by defining the service’s output to be any response produced by a majority of the server replicas. Implicit in the approach are two assumptions: First, we assume that a replica-coordination protocol exists. Second, we assume processor independence—that the individual state-machine replicas do not influence each other if executed on separate hosts in a distributed system. That is, the probability \( pr_m \) of \( m \) replicas exhibiting Byzantine behavior is approximately \( (pr_1)^m \), where \( pr_1 \) is the probability of a single replica exhibiting Byzantine behavior.

A trustworthy service must tolerate attacks as well as failures. Availability, integrity, and confidentiality are typically of concern. The approach outlined above is thus seriously deficient:

- Confidentiality is not just ignored, but \( n \)-fold replication actually increases the number of sites that must resist attack because they store copies of confidential information. Even services that do not operate on confidential data per se are likely to store cryptographic keys (so responses can be authenticated). Because these keys must be kept secret, support for confidentiality is needed even for implementing integrity.
- Any vulnerability in one replica is likely present in all, enabling attacks that succeed at one replica to succeed at all. The independence assumption, manifestly plausible for hardware failures and many kinds of software failures (such as Heisenbugs), is thus unlikely to be satisfied once vulnerabilities and attacks