10. Towards a Practical Approach to Confidential Byzantine Fault Tolerance

Jian Yin, Jean-Philippe Martin, Arun Venkataramani, Lorenzo Alvisi, and Mike Dahlin
The University of Texas at Austin

10.1 Introduction

As the world becomes increasingly interconnected, more and more important services such as business transactions are deployed as access-anywhere services – services that are accessible by remote devices through the Internet and mobile networks. Such services often must access confidential data to provide service. For example, an online bank service must access a user’s checking account to process an online transfer request. In such a scenario, guarantees of availability, integrity, and confidentiality are essential. By availability, we mean that services must provide service 24/7 without interruption. By integrity, we mean that services must process clients’ requests correctly. By confidentiality, we mean that services must restrict who sees what data. Given today’s economics and technology that make it infeasible to rigorously test and verify complex components, it is more attractive to allow untrustworthy components to be assembled into a trustworthy system. A traditional Byzantine fault-tolerant (BFT) system runs different implementations of the same service on several replicas and ensures that correct computation is performed by enough correct replicas to mask incorrect replicas [10.1, 10.7, 10.8]. Recent research has shown that BFT systems can be practical for several important services as they can be implemented with low overhead compared to the unreplicated services [10.3].

Although traditional BFT systems improve availability and integrity through redundancy, existing BFT architectures make such systems more prone to compromising confidentiality. In a traditional BFT system, replicas send all replies directly to clients. Thus, if a hacker manages to compromise one of the replicas, he can steal confidential data. Moreover, in traditional BFT systems, there is a fundamental tradeoff between increasing availability and integrity on one hand and strengthening confidentiality on the other. BFT systems use several replicas to provide availability, based on the reasoning that the replicas are different and it is therefore unlikely that they all fail simultaneously. However, because it is sufficient for an attacker to compromise a single replica, this approach also increases the chance that at least one replica contains an exploitable bug, allowing the attacker to gain access to the confidential data that the service uses.

This paper discusses how to use redundancy to simultaneously improve availability, integrity, and confidentiality. We call this problem Confidential BFT (CBFT) and propose the Privacy Firewall architecture to solve it. Service replicas connect to the Privacy Firewall...
Firewall and can send messages to the outside world only through it. The firewall runs a majority voting algorithm just like the clients of a traditional BFT system and filters out faulty messages that may contain confidential data. This approach can improve the security of any replicated service. Moreover, the Privacy Firewall can be adapted for different services with little modification, thus amortizing the development cost.

The firewall system has to be correct to provide confidentiality. Even though the firewall is simple, building a formally verified bug-free firewall may not be feasible. However, redundancy can be used to improve the robustness of the firewall. Such a firewall system consists of a group of nodes that are interconnected such that any path from a service replica to the outside world is longer than a threshold \( f \). Thus, as long as there are \( f \) or fewer faulty firewall nodes, any communication from any service replica to the outside world must go through at least one correct node. Moreover, a correct node in a firewall chain can independently ensure that a unique sequence of replies results from a sequence of requests just as if this sequence of requests were processed by a single correct server. Thus, faulty machines are prevented from using steganography to leak confidential data.

In summary, this note introduces the problem of confidential Byzantine-fault tolerance and presents a system, the Privacy Firewall, to show that CBFT can be solved. The key challenges ahead are to evaluate this new design and to develop a deeper understanding of the range of solutions to the CBFT problem.

10.2 State of the Art

Previous work in Byzantine fault tolerance falls into two categories: i) using client voting to improve availability and integrity without considering confidentiality and ii) using secret sharing to improve confidentiality for a limited class of services where servers perform few or no operations on the data they store.

Redundancy techniques to improve availability and integrity without considering confidentiality have been extensively studied. The common characteristic of such systems is that all service replicas send replies to clients and voting takes place at clients. There is a significant body of work on BFT quorum systems [10.7] in which a group of service replicas use intersection properties of quorum sets to guarantee that clients retrieve values of variables written previously. Byzantine fault tolerance for arbitrary services captured by state machines have been studied in both theoretical and practical settings [10.1, 10.3, 10.8].

When considering confidentiality, the most direct approach is to have the clients encrypt all data before sending it to the servers for storage. If the servers do not have the key then the data is stored securely, even if some servers are compromised [10.5]. Although this approach provides fault-tolerant confidentiality, the only two operations that the servers can implement are “store” and “retrieve”. Threshold cryptography [10.4, 10.9] allows a threshold number of servers to generate signatures cooperatively. Less than the threshold number of faulty nodes can neither generate a valid signature nor reconstruct the shared private key.

Byzantine fault-tolerant confidentiality can theoretically be implemented using secure multi-party computation [10.2]. Secure multi-party computation guarantees that no