# Allied Agreement with Threshold Cryptography

Robert Burgess

## Abstract

Ally is a framework for building distributed services in a federated architecture. In a federation, nodes may occupy multiple, independent administrative domains, with complex trust relationships, connected by an asynchronous, wide-area network. We present a library that implements Byzantine fault-tolerant agreement, a critical building-block service in robust distributed systems, targeted at the federation model; threshold cryptography improves the basic agreement protocol with proactive, verifiable security and improved resilience to continuous attack.

## 1 Introduction

Although cloud computing services, such as those developed by Google and Amazon, are often designed to handle nodes that crash, in general enterprise services assume absolute trust and synchrony; nodes that can behave arbitrarily can cripple or subvert such services. Furthermore, they generally assume extremely low-latency links even between nodes that function precisely as expected. For smaller organizations, such as schools, to provide themselves the advantages of a cloud architecture under their smaller budgets, they must cooperate—each supplying what nodes they can, as in Planetlab or other federated systems. Indeed, the Internet itself is a federation of mutually untrusting autonomous systems that have formed a coalition to provide a routing and message-passing service.

In a federation setting, the participating nodes are in independent administrative domains, so they can not have the same level of trust in each other as can be assumed if they were on the same rack, behind a firewall; they are also likely to be separated from each other by a wide-area network, so they can also not assume consistent, low-latency, synchronous links. Indeed, links are likely to have widely varying latencies and might even be controlled by an adversary intent on subverting the correct functioning of the service. The problem, then, is to enable nodes to cooperate to perform distributed computation or provide a cloud service, without relying on mutual trust or strong assumptions regarding the network or adversary.

We present Ally, a framework for federated application development that enables distributed services to easily guarantee Byzantine fault-tolerance, by providing an agreement (or consensus) protocol, a building block for higher-level services. We employ verifiable threshold cryptography throughout the implementation to efficiently create and check statements signed by by multiple peers, which improves scalability, enables attack recovery without key revocation, and makes proactive security possible.

## 2 Related Work

Perhaps the most famous agreement protocol is Paxos, first introduced by Leslie Lamport [15, 14], which re-popularized agreement as a basic building block of reliable distributed services. Agreement itself underlies many distributed system techniques, such as state machine replication [20].

Although Paxos assumes fail-stop participants and synchronous communication, its popularity has led to Byzantine fault-tolerant variants, asynchronous variants

Ally is directly inspired by work on distributed computing frameworks, such as Google's Chubby [2] lock service, which is internally based on a Paxos implementation, and Boxwood [16], which both provides a locking service and explicitly exposes the un-

derlying Paxos agreement service. Both Google and Boxwood provide additional building blocks, such as Google's GFS [10] and Bigtable [3] storage services and MapReduce [4] distributed computation service, and Boxwood's lower-level storage and data structure abstraction services. Neither set of building blocks, however, is designed for federated systems; both depend on honest nodes, connected within a single data center for beneficial network guarantees. Additionally, Ally is open source, unlike the previous systems but like the Hadoop[1] family of distributed services, such as Zookeeper and an open-source MapReduce implementation, which also all assume trusted nodes.

Threshold cryptography [7, 6, 5, 8] can be applied to distributed system-building, as in COCA [22], in order to achieve both scalability and improved security properties such as proactive security [13, 12, 11, 9], which provides the ability to defend against attacks by mobile adversaries [19], which attack, compromise, and control nodes for limited periods before moving on adaptively. The Ally implementation employs a trusted dealer to set up the threshold cryptosystem; however, there exist protocols for distributed key generation, such as that of Boneh and Franklin [1, 17], which removes the trusted dealer and ensures no entity ever knows the secret information.

# 3 Design

The Ally consensus protocol is an implementation of Fast Asynchronous Paxos [18] (FaB Paxos), with signatures performed using threshold cryptography. In threshold cryptography, the signing key is split, using secret sharing techniques, among the nodes, such that a threshold of them can cooperate to produce a correct signature, but any fewer gain no information about signatures or the key itself. Ally uses Shoup's threshold RSA scheme [21], which has the benefits that nodes can separately produce signature shares, without communication, enabling asynchrony; the signature shares can then be verified independently before combining them into a full signature, providing a way to catch cheaters.

[1] http://hadoop.apache.org/

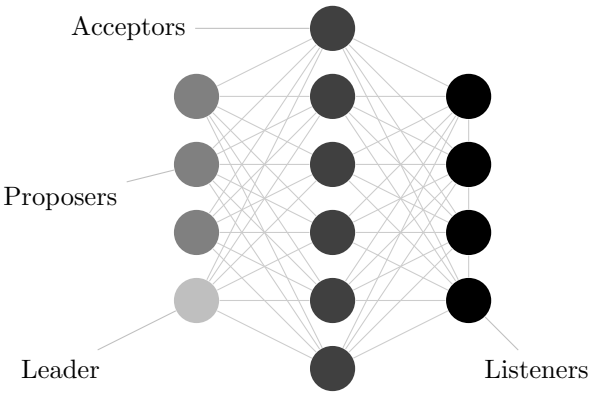


Figure 1: **Ally Agreement Architecture.** In a system that tolerates $f$ Byzantine failures, there are $3f + 1$ proposers (one of which is designated the leader), $5f + 1$ acceptors, and $3f + 1$ listeners. The processes may be colocated, so only $5f+1$ actual systems are required. The clients are separate, and the overhead of the agreement architecture can be used to support many applications.

The FaB Paxos algorithm itself is a Byzantine fault-tolerant agreement protocol, which is optimized for the common case when there is a correct leader recognized by all of the correct nodes, and the network behaves synchronously; when these assumptions fail, its performance degrades gracefully while guaranteeing correctness. It is often possible to terminate in just two rounds, the optimum for Byzantine fault-tolerant agreement.

# 4 Implementation

Ally is implemented as an open source ANSI C library, `libally`, providing the agreement protocol functionality, and also exposing an interface to distributed cryptography directly. The codebase employs techniques for simple object-orientation so that, for example, the agreement protocol can be parameterized on the distributed cryptosystem and be used with more than just the current Shoup scheme. Being written in plain C and packaged with Gnu Au-

totools provides maximum portability. The library depends on the Gnu Multiple Precision (GMP) library for mathematics, which provides excellent performance. All of the message passing is handled by the application, leaving the library itself agnostic to transport. Concrete applications can use UDP, TCP, or any other communication layer that is convenient. The library comes with a simple chat program that exercises it, as well as programs for unit testing and benchmarking.

Some features of the abstract designs are not implemented. For instance, the agreement protocol does not contain the sub-protocol to recover from a corrupt leader, and therefore no leader election. The Shoup cryptosystem offers the ability to verify individual shares in order to determine, when an overall signature is bad, what peers to exclude; this feature is not provided in the library.

The code base is available under the BSD license.

# 5 Evaluation

Because the services in Ally are intended to be the building blocks of higher-level services, their performance is critical. I will first measure the latency of the agreement protocol under ideal circumstances, that is, when the network behaves synchronously and all nodes are honest and correct.

To measure performance with crashed nodes, the experiment will be repeated with varying numbers of nodes "crashed" (not participating). To measure performance under asynchrony, the experiment will be repeated for varying random delay and loss artificially introduced in the network. Finally, nodes will approximate Byzantine failures by sending well-formed but random messages (essentially, fuzz testing), and I will repeat the experiment for varying numbers of such "Byzantine" nodes. Note that the cases in which the leader is among the crashed or failed nodes will cause different protocol paths to be followed, and I will measure them separately.

# 6 Conclusion

Cloud computing is a relatively new way of looking at distributed systems; Ally is equally applicable to other buzzwords—such as grid computing, volunteer computing, and peer-to-peer computing—when mutually untrusting participants must cooperate. The common thread is a need for complex trust relationships, enahnced fault-tolerance, and unreliable network connections. Ally provides Byzantine fault-tolerant distributed building blocks, with weak assumptions regarding the network and adversary, based on distributed cryptography for improved scalability and security.

# References


[1] Dan Boneh and Matt Franklin. Efficient Generation of Shared RSA Keys. *Journal of the ACM,* 48:702–722, July 2001.

[2] Mike Burrows. The Chubby Lock Service for Loosely-Coupled Distributed Systems. *OSDI '06: Proceedings of the 7th symposium on Operating systems design and implementation,* pages 335–350, USENIX Association, Berkeley, CA, USA, 2006.

[3] Fay Chang, Jeffrey Dean, Sanjay Ghemawat, Wilson C. Hsieh, Deborah A. Wallach, Mike Burrows, Tushar Chandra, Andrew Fikes, and Robert E. Gruber. Bigtable: A Distributed Storage System for Structured Data. *OSDI '06: Proceedings of the 7th USENIX Symposium on Operating Systems Design and Implementation,* pages 15–15, USENIX Association, Berkeley, CA, USA, 2006.

[4] Jeffrey Dean and Sanjay Ghemawat. Mapreduce: Simplified Data Processing on Large Clusters. *Commun. ACM,* 51(1):107–113, ACM, New York, NY, USA, 2008.

[5] Yvo Desmedt. Some Recent Research Aspects of Threshold Cryptography. *Information Security,* Tatsunokuchi, Ishikawa, Japan, September 1997.

[6] Yvo Desmedt and Yair Frankel. Threshold Cryptosystems. *Advances in Cryptology,* pages 307–315, 1989.

[7] Yvo Desmedt and Yair Frankel. Shared Generation of Authenticators and Signatures (extended Abstract). *CRYPTO '91: Proceedings of the 11th Annual International Cryptology Conference on Advances in Cryptology,* pages 457–469, Springer-Verlag, London, United Kingdom, 1992.

[8] Yair Frankel and Moti Yung. Distributed Public Key Cryptosystems. *PKC '98: Proceedings of the First International Workshop on Practice and Theory in Public Key Cryptography,* pages 1–13, Springer-Verlag, London, United Kingdom, 1998.

[9] Yair Frankel, Peter Gemmell, Philip D. MacKenzie, and Moti Yung. Optimal Resilience Proactive Public-Key Cryptosystems. *IEEE Symposium on Foundations of Computer Science,* pages 384–393, 1997.

[10] Sanjay Ghemawat, Howard Gobioff, and Shun-Tak Leung. The Google File System. *SOSP '03: Proceedings of the nineteenth ACM symposium on Operating systems principles,* pages 29–43, ACM Press, New York, NY, USA, 2003.

[11] Amir Herzberg, Markus Jakobsson, Stanislaw Jarecki, Hugo Krawczyk, and Moti Yung. Proactive Public Key and Signature Systems. *ACM Conference on Computer and Communications Security,* pages 100–110, 1997.

[12] Amir Herzberg, Stanislaw Jarecki, Hugo Krawczyk, and Moti Yung. Proactive Secret Sharing Or: How to Cope with Perpetual Leakage. *Lecture Notes in Computer Science,* 963:339–352, 1995.

[13] Stanislaw Jarecki. Proactive Secret Sharing and Public Key Cryptosystems. Master's Thesis, Massachusetts Institute of Technology, Cambridge, Massachusetts, 1995.

[14] Leslie Lamport. Paxos Made Simple. *ACM SIGACT News,* 32(4):18–25, December 2001.

[15] Leslie Lamport and Keith Marzullo. The Part-Time Parliament. *ACM Transactions on Computer Systems,* 16:133–169, 1998.

[16] John MacCormick, Nick Murphy, Marc Najork, Chandramohan A. Thekkath, and Lidong Zhou. Boxwood: Abstractions as the Foundation for Storage Infrastructure. *OSDI'04: Proceedings of the 6th conference on Symposium on Opearting Systems Design & Implementation,* pages 8–8, USENIX Association, Berkeley, CA, USA, 2004.

[17] Michael Malkin, Thomas Wu, and Dan Boneh. Experimenting with Shared Generation of RSA Keys. *Network and Distributed System Security Symposium,* San Diego, California, 1999.

[18] Jean-Philippe Martin. Fast Byzantine Consensus. *IEEE Trans. Dependable Secur. Comput.,* 3(3):202–215, IEEE Computer Society Press, Los Alamitos, CA, USA, 2006.

[19] Rafail Ostrovsky and Moti Yung. How to Withstand Mobile Virus Attacks. *ACM Symposium on Principles of Distributed Computing,* pages 51–59, Montréal, Canada, August 1991.

[20] Fred B. Schneider. Implementing Fault-Tolerant Services Using the State Machine Approach: A Tutorial. *ACM Comput. Surv.,* 22(4):299–319, ACM, New York, NY, USA, 1990.

[21] Victor Shoup. Practical Threshold Signatures. *Lecture Notes in Computer Science,* 1807, 2000.

[22] Lidong Zhou, Fred B. Schneider, and Robbert van Renesse. COCA: A Secure Distributed on-Line Certification Authority. *ACM Transactions on Computer Systems,* 20(4):329–368, November 2002.