Senior Project Proposal:
Collective Consensus

Dylan Visher

January 17, 2015

1 Overview

This semester I will be working on my senior project as part of the Dissent Group at Yale. This group includes Principle Investigators: Bryan Ford, Joan Feigenbaum, Vitaly Shmatikov; Project Learners: David Wolinsky, Henry Corrigan-Gibbs, Amir, Houmansadr, Ramakrishna Gummadi; as well as PhD students and other collaborators. I will be working under Professor Bryan Ford with other Yale undergrads on the Dissent project.

Overall the Dissent builds on previous knowledge of DC nets, and verifiable shuffle algorithms, in an effort to provide strong anonymity guarantees. As it is building on these secure but traditionally unscalable anonymity technologies, there is an ongoing effort to deal with scaling the system in a decentralized fashion, while maintaining the same security guarantees and not sacrificing too much performance.

This semester I will specifically be working on the Dissent Collective Consensus subproject, specifically a Collective Signing protocol using ElGamal signatures. Collective signing is an imperative piece of a reliable directory service. Dissent will have a directory service similar to that of Tor, maintaining a reliable, trustworthy list of all participating Dissent participants. The Dissent directory service however will remain decentralized over a variety different nodes. Collective Signing is a protocol, in which, many individual nodes can contribute to produce one larger digital signature. This Collective Signing protocol can then be leveraged to become the basis for other important protocols like a Decentralized Timestamping Service, an Agreed-Upon Log, and a means for Dynamic Configuration. All these other protocols are based off of Collective Signing’s ability to produce a consensus regarding a single value across all nodes efficiently.

Though I will be working with other undergrads on this specific project, as it is so large, we will be working on separate pieces of the Collective Signing process. And as the semester develops we will expand into working on other disjoint pieces of the Collective Consensus protocol dependent on this piece, as far as time permits.

2 Collective Consensus

Initially we will build a basic version of the Collective Consensus protocol under the assumptions that all participating Dissent nodes form a rooted tree and that nodes never become unavailable. Only after this basic version is complete will we make this tree structure dynamically self-organizing and able to handle node unavailability.

The protocol operates over the course of several rounds, producing one log entry per round, much like the Multi-Paxos protocol. Each log entry contains: new data for this round, a "history" of previous log entries, and an ElGamal signature verifying this new log entry. The ElGamal signature is generated by all participating nodes, using all their public keys.

2.1 Abstract Network Tree

The first stage of this collective signing protocol is abstracting the network stack away. Ideally we want
Dissent to work over a variety of network and data protocols that can be user defined. Since Dissent organizes host-nodes in an implicit tree structure to help minimize network traffic, we will provide an abstract interface for node communication in this tree structure.

Initially we will assume that this tree structure is already in place and that each node has knowledge of the configuration. This abstract interface will then be implemented by several different types of host nodes, for testing and real world purposes.

One of the principle host node types will use coroutines and channels for communicating amongst an in-memory version of Dissent for testing purposes. Other host node types will be implemented for different network communication protocols like TCP.

We will also build a system for reading JSON configuration files. This system will be responsible for building and testing this tree structure using the coroutine-channel Host mentioned above. This work will then be expanded to interface with the python NetworkX module. We will construct random graphs of different topologies and test the performance and behavior of the system. Specifically we will look at: random regular graphs, Watts Strogatz graphs, Barabasi Albert graphs, Powerlaw Cluster graphs, Navigable Small World Graphs, and Waxman graphs. This will give us access to many planar and non-planar topologies that are representative of different real world scenarios.

2.2 Elgamal Signing

Once this tree structure has been established, we will build the protocol for constructing a single ElGamal digital signature from the public keys of the participating nodes.

This Collective signing protocol takes place over two 'root-leaf-root' passes, which can be viewed in four stages, or as two RPC requests initiated from the root of the tree. The "RPC call" Announce is initiated by the root of the tree and elicits a Commit response. After all commits have been aggregated by the root, it issues a Challenge call to which all node give back their Response. The general flow of these "RPC calls" is to forward the RPC call down to the children of the receiving node, aggregate the responses, process these responses and then return the new aggregated response up the tree.

**Announce: Call** The Announcement call initiates a new round. It is initiated by the root of the tree, the "proposer". All other nodes, "acceptors", in the tree forward the Announcement to its immediate children.

**Commitment: Response** After forwarding the Announce message down to its children, each intermediate node waits for the commitment responses of the children. This node then adds its own contribution for the round, and passes it up.

**Challenge: Call** After receiving the collective ElGamal signature from the rest of the tree, the root sends down a hash of this aggregated commitment such that each child can verify efficiently that its contribution was correctly included.

**Response: Response** Each child node computes its part of the ElGamal signature response and combines it with that of its children. This is then passed up to the root and can be verified against the top-level composite public key.

After the response phase, the generated ElGamal signature is broadcast to all nodes, where it can be verified if desired.

3 Building on Collective Signing: Next Steps

The result of the Collective Signing protocol described above is a verifiable consensus. By the end of the ElGamal signing process, the nodes have reached a consensus on the commitments of the given round. This consensus protocol will be iterated over many rounds, and will be utilized for a decentralized timestamp, logging, and configuration service for Dissent.
3.1 Distributed Log

We will build upon the ElGamal signing first by using it for decentralized logging. One of the central problems in many distributed systems is that of consensus. In Dissent rounds, every node will contribute to the collective log during the commitment stage of the Collective-Signing. In addition to adding its signature to the partial commits of the nodes below, it will also add its log contributions for the round. These log contributions will be aggregated in a verifiable Merkle tree structure. The result of a collective signing round now is a verifiable log in the form of the top-level aggregate Merkle tree. The fact that all nodes have signed this Merkle tree on the way up means every node can verify that their log contribution was included in the round.

3.2 Dynamic Configuration

In order to allow for groups where the membership is dynamic, the configuration can be handled, similar to the log, by the evolution of a collective Merkle "configuration" tree. This tree will contain the public keys of all participants, the aggregate subtree key for each subtree within the top-level tree, and the number of descendants of each node. If a child fails the parent can produce an exception record that includes the appropriate records in the agreed upon Merkle configuration tree. These exceptions can then be easily verified by any other participant.

3.3 Acceptor Liveness

The problem with the scheme mentioned above is that Collective Signing is dependent on all participants to be present for the collective log signing. However, in practice, this is not a good assumption to make. We will be testing two different approaches to solving the participant availability problem for Collective Signing.

Since participants in this case are servers that can be pruned by the community (i.e. only allowed in after demonstrating provisioning and availability), it might be safe to assume that though failures do happen, most failures are rare and brief. With this in mind we can alter the Collective Signing process such that if a node discovers that one of its immediate children is temporarily offline or unresponsive, the parent leaves its descendants out of the collective signature-generation process. This will result in a modified, not ideal, aggregate public key. This parent node will pass up its commitment response with a list of exceptions, for each node whose contribution is missing. This exception list will be verifiable by any other node in the tree. The aggregate key that is generated will be verifiable against the subset of public keys that participated in the round, which can be reconstructed knowing the configuration and the exceptions list.

A different subgroup of will be working on the problem through Signing Key Sharing. In this case, Shamir secret sharing will be used to split the failed participant’s key into shares among trustees. Then when the participant fails, the other nodes, all working together, can reconstruct the server’s key and participate in rounds for it.

4 Personal Contribution

As a member of the Yale DeDis Research group my contributions will always be in the context of the group. That said, I will be working on my own pieces of this larger whole. I will be contributing to the Github Dissent library at [https://github.com/dedis/prifi](https://github.com/dedis/prifi), specifically on the diss/prifi/coco library. I will be leading the effort on the decentralized network setup required for ElGamal signing in the Collective Signing part of Dissent. I will also be participating, though in a much more minor role, in the Collective Signing protocol itself. As the semester continues my work will evolve into the Dynamic Configuration problem for Collective Consensus. This work will include flushing out the details of the Dynamic Configuration problem as well as implementing it on top of the Collective Signing protocol that will be established earlier in the term.