Fast Anonymous Consensus and Private Authentication in Large Distributed Systems

Vivek Nair, University of California, Berkeley
Bolton Bailey, University of Illinois at Urbana-Champaign

***Abstract*—We present a simple decentralized protocol for achieving all-to-all broadcast consensus in large permissioned distributed systems while providing strong privacy for every participant. Our method is optimized for real-world performance by limiting the use of anonymous broadcasting to only an initial input phase and then allowing the entire agreement phase to take place over fast open networks. We demonstrate an important application of our protocol in establishing a pseudonymized (or "shadow") public key infrastructure in any authenticated setting. We performed a large-scale demonstration with 1024 AWS t2.nano EC2 instances and found that 128 parties can achieve consensus on 128 anonymous inputs in about 43 seconds and can construct a shadow PKI in about 46.5 seconds. In a simulated private group chat application with 256 users sending 128 messages each, the shadow PKI provided over 36 times better performance than ring signatures alone with over 99.99% probabilistic security and privacy against a 65% adversary.**

I. Introduction

A. Overview

The design of current broadcast consensus (reliable broadcast) protocols for authenticated settings seems to be fundamentally at odds with notions of privacy and anonymity. They require participants to communicate over an open network and attach digital signatures to their messages, and need the identity of a fixed sender to be known to all participants ahead of time. In this paper, we explore why anonymous consensus is a harder problem than it might initially seem, and why simple solutions for anonymizing existing consensus algorithms fail to produce efficient anonymous protocols. We propose a two-phase approach to anonymous consensus and demonstrate its performance and privacy benefits. We then explore an important application of anonymous consensus in pseudonymizing public key infrastructures.

B. Shadow PKIs

Distributed systems are traditionally divided into authenticated and unauthenticated (or permissioned and permissionless) settings. In an authenticated setting, an established PKI is present such that all parties are known to each other and can validate messages from other parties via the use of cryptographic signatures. In an unauthenticated setting, parties are not known to each other, and have a degree of privacy, but at the cost of message authentication and Sybil resistance mechanisms not being available (typically giving rise to proof-of-work based mechanisms for consensus).

In this paper, we propose a third, pseudonymously authenticated setting, made possible by fast anonymous consensus. In this setting, parties reach consensus on a "shadow" (pseudonymized) PKI, whereby messages can be verified as belonging to one of the parties in the shadow PKI without being traceable to a specific process[1] while maintaining Sybil resistance. We accomplish this by providing a mechanism for transforming any PKI into a shadow PKI while preserving critical security properties. Using this transformation, any distributed system designed for an authenticated setting can be run in a pseudonymously authenticated setting, providing the benefits of message authentication and Sybil resistance with the privacy of an unauthenticated system.

C. Applications

The core technique of our paper is the anonymization of existing consensus protocols. Consensus protocols are an essential part of distributed systems which allow parties to reach consensus on the state of a system. Various consensus protocols exist, each adapted to a particular setting [6], [9], [2]. Our paper allows these protocols to be executed anonymously. An important application of anonymous consensus is PKI pseudonymization, which itself has important implications for security and privacy.

While any authenticated distributed system can be anonymized using our techniques, the most interesting applications of this technology involve decentralized systems where security and privacy are both critical. Anonymous payments systems where the participants belong to a set of authorized users are another use case. Anonymous communication systems allowing parties to anonymously message a set of trusted users are a third example, which we will follow closely throughout this paper. In general, our anonymous PKI system enables applications that require a group of authorized users to be able to interact anonymously within the group while excluding unauthorized external users.

Further applications of this technique involve improving the privacy and anonymity of participants in a secure multiparty

[1]In this paper, we use the terms nodes, parties and processes synonymously to describe a participant in a distributed system. We similarly use the terms Byzantine, faulty, malicious, and dishonest synonymously to describe nodes which deviate from a protocol.

Network and Distributed Systems Security (NDSS) Symposium 2022
23-26 February 2022, San Diego, CA, USA
ISBN 1-891562-61-4
https://dx.doi.org/10.14722/ndss.2022.23xxx
www.ndss-symposium.org

computation scheme. We discuss this application further in the future work section.

Another potential application of PKI anonymization is e-voting. The possibility of allowing voting to be carried out between computers has been studied by the cryptography community [15]. With a shadow PKI, parties can carry out a vote anonymously by signing their votes with a pseudonymous key.

D. Related Works

Our protocol works on the basis of an anonymous broadcast system. Our implementation uses a Dandelion-based broadcast [23], but in principle any synchronous broadcast protocol with privacy preserving properties could be used. To this point, a variety of mix network architectures [24] could serve as drop-in replacements.

Another potential design to accomplish the distribution of a shadow PKI is to use a blockchain-based method. This method would work by having the participants instantiate a blockchain, represent keys as tokens on the chain, then transfer tokens to other accounts they control in an anonymous way. To preserve anonymity, this transfer could use a protocol such as [25]. Our approach avoids the overhead of a blockchain by reaching consensus on a new PKI directly. Another benefit of avoiding blockchain-based methods is to sidestep constraints on the resistance to Byzantine attacks. In principle, our protocol can tolerate any number of Byzantine nodes, as long as it is instantiated with a suitably strong consensus protocol. Our protocol does not rely, for example, on assumptions about the computational work available to an adversary [26].

Several other works have studied the idea of "anonymous distributed systems" under different security models than ours:

Ciaraldi et al. [18] describe risks that arise in the practical application of anonymous distributed systems including the possibilities of corrupted data and malicious code. We deal with the risks that this paper proposes by basing our protocols on cryptographic constructions, which we assume are correctly implemented.

Hendrickx and Tsitsiklis [8] explore the limitations of a certain class of anonymous system. In contrast to our work, this paper focuses on a form on anonymity where nodes are unaware of their place in the network and only have a broadcast protocol to communicate with their neighbors. Instead, we assume that there are a fixed set of participants in our protocols, and that all processes are connected by channels that allow them to broadcast anonymously to everyone.

Haenni and Spycher [27], Kaufmann [28], Critchlow and Zhang [4], Choi et al. [3] and Wang et al. [14], study systems for the anonymous distribution of certificates. These works deal with systems having a certificate authority to assist in the distribution of these certificates. By contrast, our work deals with permissioned, but decentralized protocols for key distribution, which we use newer cryptographic protocols to achieve.

E. Contributions

In this paper, we offer a few contributions to the practice of anonymous distributed systems. Our primary contribution comes in the form of an anonymous multi-value all-To-all broadcast consensus protocol. This protocol allows a group of parties, using a ring signature scheme, to distribute to each other a list of values without identifying which values originate with which party or allowing malicious parties to introduce more values than the protocol specifies. This protocol is designed to minimize the use of covert broadcasts and ring signatures - to accomplish this, we adopt a two phase protocol with an initial covert broadcast ("input") phase, followed by an agreement phase.

We also introduce protocols for transforming PKIs: A PKI Reconstitution Protocol which allows parties to switch signature schemes, and a Shadow PKI Protocol, which uses the reconstitution protocol in conjunction with the anonymous multi-value all-To-all broadcast consensus to obtain a PKI where individuals identities cannot be associated with their public keys.

We provide a systematization of the definitions of security behind all of these protocols, and make rigorous proofs for these properties in the appendix.

Finally, we justify our constructions by creating a demonstration of their performance. We run our demonstration on AWS and relay the results of this experiment, which show speed improvements over a naive approach on a simulated anonymous chat application. The code for our implementation is open source and can be found on GitHub (link to be provided in the published manuscript).

II. Anonymous Broadcast Consensus

A. Overview

Consider a distributed system with n parties, of which $f < n$ may be faulty (Byzantine) and the remaining parties are honest. In a single instantiation of the broadcast consensus problem, one of these parties is designated the "sender". The known sender inputs a chosen value which is to be received by the honest parties. The broadcast consensus problem requires that:

1) Liveness: all honest parties eventually output a value
2) Validity: if the sender is honest, its value is the decision value
3) Agreement: all honest parties decide the same value

Broadcast consensus algorithms, per their present definition, are useful for a variety of important applications in the distributed systems space. They are not, however, useful for applications requiring privacy or anonymity, due to the requirement for the identity of the sender to be known to all other parties, and for the sender to directly communicate its input value to one or more other parties. Both requirements necessarily make the sender's input directly linkable to their identity, providing no feasible means of anonymity. In an Anonymous Broadcast Consensus protocol, we seek to add a fourth requirement:

4) Anonymity: the identity of the sender is not revealed to any other party

Of course, given the closed nature of a permissioned distributed system, parties must still be able to verify that the sender is indeed an authorized member of the system, without learning anything further about their identity. The difficulty of achieving anonymity in current broadcast consensus protocols is exemplified by Dolev-Strong [6].

B. Dolev-Strong

Dolev-Strong is perhaps the most famous authenticated broadcast consensus protocol of all time. It provides a solution to the broadcast consensus problem satisfying properties 1-3 above with perfect provable security. The authors demonstrate a theoretical minimum of $f+1$ rounds and $O(n^2 f)$ words for solving the broadcast problem in a synchronous network against any adversary controlling f out of n parties. They further show that at least $O(n^2 f)$ signatures and $O(n^3 f)$ signature verifications are required to achieve broadcast consensus in a permissioned environment.

Dolev-Strong assumes that parties will agree on the identity of a fixed sender prior to the execution of a protocol instance, making the notion of "privacy" in this protocol somewhat meaningless. Even setting this aside, Dolev-Strong's layered signature system makes it perfectly clear to any message recipient whether a value is being received directly by a sender or is simply being forwarded by another party. The point-to-point networking used by the scheme will therefore reveal the identity of the sender to all other parties during the protocol's first phase. Further, by requiring senders to directly sign their input, the scheme not only removes any notion of privacy, but further makes exculpability and repudiation impossible.

It is clear that the anonymization of broadcast consensus requires us to shift our premise from recognizing a fixed sender to allowing the sender to be any one of the recognized parties. Care must then be taken to anonymize both the networking component of the protocol, as well as the digital signatures. While well-known techniques exist for solving both of these problems, a naive application of these solutions to existing consensus protocols is significantly problematic.

C. Straw Man Solution

We will begin by considering what we believe to be the most obvious solutions to the two anonymity problems in Dolev-Strong.

1) Network Anonymity: With respect to communications, packet networking necessarily reveals identifying information (e.g. IP address) of a message sender and recipient in all commonly used network architectures. Anonymous communication systems like Tor are specifically designed to circumvent this limitation. We may naively consider replacing all point-to-point communications in Dolev-Strong with Tor messages.

2) Signature Anonymity: We consider now the issue of authenticating a sender without revealing their identity. Ring signatures are a cryptographic primitive intended to allow any member of a group to sign a message without revealing their identity within the group. On their own, ring signatures do not even allow it to be known whether two separate signatures came from the same party, a functionality required by Dolev-Strong. So-called "linkable" ring signatures aim to solve this problem by retaining pseudonymity but allowing any pair of signatures from the same party to be linked. We would therefore naively consider replacing all digital signatures in Dolev-Strong with linkable ring signatures.

3) Performance Considerations: It is argued that the two aforementioned substitutions produce a technically correct solution to the anonymous broadcast consensus problem. However, the proposed naive solution is so unperformant as to be totally impractical for large distributed systems. Consider even a relatively small network of $n = 64$ nodes and $f = 32$ Byzantine nodes. We know that a total of $n^2 f = 131,072$ signatures, $n^3 f = 8,388,608$ verifications, and $n^2 f = 131,072$ messages would be required in this scenario. We measured the time to compute a single 2048-bit RSA signature and verification, as well as the time to compute a single and verification in the Tsang & Wei [20] traceable ring signature scheme, and found that RSA was 106x faster at signing and 6,780x faster at verifying. We further measured the latency for a point-to-point message versus a message over the Tor network and found that Tor was 4200x slower than point-to-point communications. More details about the precise experimental setup used to obtain these numbers are provided in the Experimental Evaluation section. Together with the number of operations needed for Dolev-Strong and our straw man protocol, we obtained estimated computational time for each algorithm to execute one full iteration. The results are shown in Table I.

TABLE I
PROJECTED RELATIVE PERFORMANCE OF STRAW MAN SOLUTION

	Signatures				**Messages**		**Time / Node**
	RSA Signature		**Ring Signature**		**Direct**	**Tor**	
	Sign	**Verify**	**Sign**	**Verify**			
Time (One Operation)	5.0ms	5.0ms	530ms	33.9s	0.5ms	2.1s	
Count (Dolev-Strong)	131k	8.4M	0	0	131k	0	
Time (Dolev-Strong)	10.9m	11.7h	0	0	65.5s	0	11.1m
Count (Straw Man)	0	0	131k	8.4M	0	131k	
Time (Straw Man)	0	0	19.3h	9.0y	0	3.2d	51.5d

We estimate that while the network using Dolev-Strong would take 11.1 minutes to reach consensus, the naive approach for anonymous consensus would take over 51.5 days to reach the same result, or over 6680x longer. The data we provide exemplifies the true difficulty of anonymizing standard consensus algorithms. While ring signatures and anonymous communication protocols are useful and necessary tools, they carry an overwhelming performance impact, and do not constitute a "find and replace" solution if performance is a concern. A significant goal of this paper was therefore not only to provide a technique for anonymous consensus, but to do so in an efficient and practical manner, by reducing our reliance on ring signatures and covert channels to the minimum possible usage. As a preview of the fruits of this effort, our final system achieves one-to-all anonymous broadcast consensus for $n = 64$ nodes in just 11.7 minutes with guaranteed security, or to as little as 11.3 seconds for all-to-all anonymous broadcast with probabilistic ($> 99.99\%$) security.

D. Two-Phase Approach

To address the performance impact of anonymous networking and signatures, we suggest splitting the anonymous broadcast consensus protocol into two phases: an anonymous input phase, and an open agreement phase. This 2-phase technique is based on the observation that in a typical broadcast consensus protocol, only in the early stages of the protocol does a sender exhibit different behaviour to any other node. In later rounds, senders may behave identically to all other honest participants, and thereby do not risk compromising their privacy. This means that anonymization techniques such as ring signatures and covert broadcasts are only required when the sender initially provides their input, after which fast point-to-point communications and normal digital signatures may safely be used with no loss of anonymity. Since the agreement phase constitutes the vast majority of the computational cost incurred by a consensus algorithm, the two-phase approach provides a significant performance advantage. The phases are specifically broken down as follows:

1. Input (Anonymous):
   1.1. Sender generates linkable ring signature for their input value (e.g. using the Tsang-Wei [20] LRS scheme)
   1.2. Sender anonymously sends input to all other parties (e.g. using Tor)
2. Agreement (Open): All parties execute broadcast consensus (e.g. Dolev-Strong) on the received anonymous input (equivalent to starting Dolev-Strong after the 1st round)

It is important to node that Tsang-Wei is not the only suitable ring signature scheme, nor is Tor the only suitable anonymous communication protocol. In fact, Dolev-Strong is just one of many broadcast consensus protocols that could be used in the agreement phase of the proposed anonymous broadcast consensus system. We therefore consider all of these primitives or ("building blocks") of our anonymous broadcast scheme to be modular and replacable according to the implementor's needs. In fact, in our Experimental Evaluation section, choose different protocols for all three primitives and discuss our reasons for doing so. Furthermore, we claim, and later demonstrate, that the technique presented here will never be less secure than the chosen primitives. For example, it provides guaranteed security if Dolev-Strong is used, while guaranteeing 99.5% probabilistic security if a gossip protocol with 99.5% security is used.

Considering the $n = 64$ nodes and $f = 32$ Byzantine nodes use case, reducing linkable ring signature and covert channel usage to the input stage allows only the first round to require the use of these techniques, while the remaining 32 rounds can still take place over open channels and with fast signatures. This improvement is critical for bringing anonymous consensus into the realm of practicability.

E. Magnitudes of Consensus

While we have now presented a practical one-to-all broadcast consensus scheme, this is actually the bare minimum for what we consider useful for most real-world applications. In reality, there are many orders of scale for broadcast consensus. Although no standardization was readily available, we would charactarize the degrees of anonymous broadcast consensus as follows:

1. One-to-All Broadcast Consensus $\leftarrow$ *We are here*
2. Single-Value All-to-All Broadcast Consensus
3. Multi-Value All-to-All Broadcast Consensus

1) Single-Value All-to-All Broadcast Consensus: In fully decentralized systems with no specially privileged nodes, 'degree 2' (all-to-all broadcast) consensus is usually what is required for practical use cases. In such systems, most parties will require the chance to provide input, whereas one-to-all broadcast consensus makes more sense in the context of a single privileged sender. For example, in an electronic voting application where each node in a distributed system has an anonymous vote that must be visible to all other nodes, all-to-all broadcast consensus is required. This particular example illustrates the need for another important property in all-to-all broadcast consensus: Sybil resistance. Specifically, for an anonymous single-value all-to-all broadcast consensus algorithm, it is imperative that no party should be allowed to have more than one input recognized by the network (thereby allowing them to vote more than once).

2) Multi-Value All-to-All Broadcast Consensus: Continuing now with the example of voting, we observed that most voting scenarios will involve multiple independent selections (e.g. multiple offices in an election, or multiple proposals in a board meeting). This can of course be accomplished by including all of one's selections in a single message using a single value all-to-all consensus protocol, but this will have the effects of linking their votes. A multi-value consensus protocol extends a single value all-to-all anonymous broadcast by allowing each party to provide multiple values (but still no more than one per "issue" at hand), with those values being not only anonymous but also unlinkable to each other.

F. Protocol Reductions

Once again, trivial (but non-optimal) solutions exist that could allow one to simply reduce each protocol to a simpler variant of itself. An all-to-all protocol could conceivably be constructed by invoking a one-to-all broadcast n times, with each party having one chance to be the designated sender. A multi-value all-to-all protocol can similarly be constructed by invoking a single-value all-to-all protocol m times for the m separate issues at hand. If a multi-value all-to-all Anonymous Consensus Protocol were indeed constructed this way, it would require $n * m$ separate invocations of both the input and agreement phases of the one-to-all consensus protocol. As we saw earlier with our straw man one-to-all anonymous solution, these are "solutions" only in technicality, and provide no meaningful performance improvement over the one-to-all consensus protocol.

We note, however, that because agreement is fundamentally required on just the end result of the consensus protocol, we can get away with repeating only the input phase $n * m$ times for multi-value all-to-all consensus, with still just a single invocation of the agreement phase (on the entire result of the input phase) regardless of the number of senders or the number of independent values.

What follows is the efficient extension of the previously presented fast one-to-all anonymous broadcast consensus protocol into a fast multi-value all-to-all anonymous broadcast consensus protocol.

G. *Multi-Value All-To-All Anonymous Consensus Protocol*

We now outline the functioning of the multi-value all-to-all anonymous broadcast consensus protocol.

As we have mentioned this protocol works in two phases. In this initial phase, processes receive a list of input values and generate a ring signature for each element of the list. Each signature is generated with respect to a tag corresponding to the index of the input, so that honest signatures cannot be linked to each other, but separate signatures in the same tag from the same key can be linked. These signed values are then sent through an anonymous broadcast system to all other processes in the network, which, together with the properties of the ring signature, prevents nodes from learning the identities of the sent messages.

When these values are received, we enter an agreement phase where all processes broadcast all values they received in the previous phase to all other nodes over a publicly visible network. This guarantees that all nodes reach consensus on the set of all values that were sent in the first phase, which provides consistency in the outputs of the nodes.

Having a set of all values with their linkable ring signatures, nodes then use the cryptographic checks to remove any values that have evidence of misbehavior. In particular, they remove any invalidly signed messages and any messages which link to other messages, which guarantees that malicious processes cannot introduce more values than if they behaved honestly. The ability for honest nodes to always have their values included is protected by the tamper resistance properties of the ring signature. Finally, nodes output the remaining values.

We consider the protocol described in this section to be the most significant and fundamental presentation of our broadcast consensus algorithm. We note that although presented as a multi-value all-to-all constriction, it can be reduced back to a single-value all-to-all construction simply by setting $M = 1$, and can be further reduced to a one-to-all construction by having only one node provide non-null input in the input phase. In each case, the "simpler" version is executed with no worse performance than a dedicated implementation for that purpose alone. Because this protocol can be efficiently utilized to provide one-to-all, all-to-all, or multi-value all-to-all consensus, we consider it the only fundamental protocol necessary for all meaningful application scenarios.

We hope this outline provides an intuitive understanding of the claimed properties of our consensus protocol and why they should be upheld provided valid underlying primatives. Far more rigorous specifications and proofs of the claimed properties of this protocol are provided in this paper's appendices.

Armed with the versatility of an efficient multi-value all-to-all anonymous broadcast consensus protocol, we now present an important application of this method.

III. Shadow PKIs

We now shift our focus to the discussion of a practical application scenario: a pseudonymous group messaging application consisting of a closed (authenticated) set of members who wish to communicate internally under a pseudonym without revealing their identity. Suppose the parties intending to construct such a chat room have already established a permissioned distributed system with point-to-point communications between all nodes.

A. *Straw Man Solution*

A simple approach to solving the problem of a pseudonymous group chat would be for participants to use a linkable ring signature for all chat messages. This would protect a participant's anonymity within the group while still ensuring that multiple messages from the same participant are linkable to their pseudonymous identity. If a PKI does not exist with respect to an LRS scheme, the Reconstituion Protocol in part B could be used to transform the PKI. In part C, we present the "shadow PKI" approach for solving the pseudonomous messaging problem, and in part E, we compare the performance of that approach to the naive approach described here.

B. *Reconstitution Protocol*

A nuance of the way we define the Shadow PKI Protocol in our formalization (see Appendix A) is that the initial and resultant PKIs are defined with respect to the same digital signature scheme (for example, from nonymous 2048-bit RSA keys to pseudonymous 2048-bit RSA keys). This presents a challenge for the use of our consensus protocol, which specifically requires a linkable ring signature scheme, as part of the Shadow PKI Protocol. We therefore present a reconstitution protocol for use as a component of the Shadow PKI Protocol. The protocol is as follows:

1) Generate a key pair with respect to the desired output PKI
2) Sign the output public key with your input private key
3) Execute an all-to-all broadcast consensus algorithm on the signed output public keys

The PKI Reconstitution Protocol we outline here is also useful in some standalone applications. For example, the straw man solution for pseudonymous group messaging could utilize the reconstitution protocol to establish the linkable ring

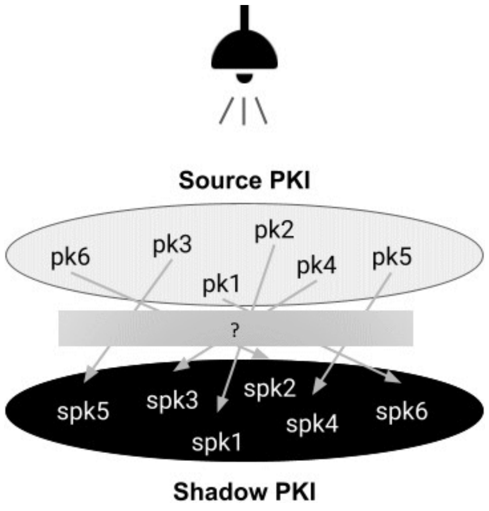

Fig. 1. Shadow PKI Protocol visualization

signature keys it uses for signing and validating the chat messages, if it is assumed that the initial PKI is established with respect to a standard digital signature scheme. It could also be used, for example, to implement periodic key rotation in a decentralized system.

C. Shadow PKI Protocol

The Shadow PKI Protocol makes use of the anonymous broadcast protocol to establish a pseudonymous PKI. The first step of the protocol is the previously described reconstitution protocol, where nodes use their existing PKI to establish a PKI in the linkable ring signature scheme. This reconstitution is done by the processes generating and sending ring signature keys and reaching consensus on them.

The next step is each process to generate keys for the shadow PKI. The processes can then use the anonymous broadcast protocol with the generated ring signatures to send their generated shadow public keys. These keys then become their own PKI, with all honest processes having a key present and with identities disassociated from the keys though the anonymity guarantee of the broadcast protocol.

We consider the PKI resulting from this process to be a "Shadow PKI," and the process itself to be a "Shadow PKI Protocol," because the process is analogous to the original PKI casting a dark shadow which consists of the same elements but where the precise mapping between keys in the source and Shadow PKI are lost in the transformation. An illustration is provided in Figure 1.

As with the consensus protocol, this outline hopefully serves to provide an intuitive understanding of the steps and claimed properties of the Shadow PKI Protocol, with more detailed specifications and proofs of the claimed properties being provided in this paper's appendices.

D. Multiple Shadow PKI Protocol

Consider, now, a pseudonymous chat application with multiple group chat rooms or "threads," in which the same group

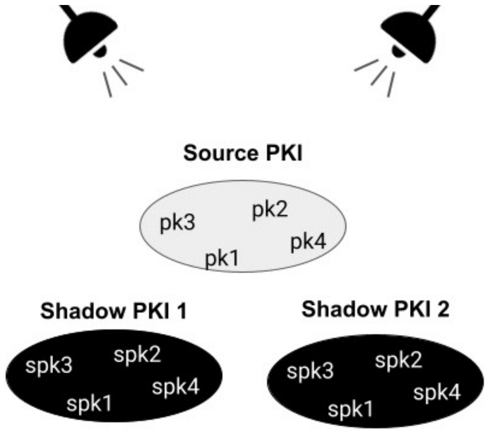

Fig. 2. Multiple Shadow PKI Protocol visualization

of users wishes to remain independently pseudonymous within each thread (such that their messages are unlinkable across separate chat rooms). What we need to do in this case is to establish multiple separate pseudonymous Shadow PKIs with linkability neither between keys in Shadow PKIs nor back to their owners' identities. We have previously introduced precisely the tool necessary to do this: anonymous multi-value all-to-all broadcast consensus. By generating multiple keys per node, and supplying them as separate inputs to our multi-value consensus algorithm, we obtain a multiple Shadow PKI Protocol which outputs several independent shadow PKIs from the same source PKI. We consider this analogous to multiple light sources causing a single object to cast multiple shadows (illustrated in Figure 2). Using exactly one of the resulting PKIs for each chat room achieves the desired effect of creating multiple pseudonymous keys for each user. The use of our efficient multi-value consensus protocol makes this process significantly more efficient than simply performing multiple independent iterations of the Shadow PKI Protocol.

E. Performance

We argue that the Shadow PKI Protocol constitutes a critical application of our Anonymous Consensus Protocol for performance reasons as well as for its unique applications. While considerable effort has been put into optimizing the performance of our consensus protocols, we remain limited by the fundamental fact that linkable ring signatures are far less efficient than standard digital signatures. The ability to apply the Anonymous Consensus Protocol a single time to obtain a pseudonymized "Shadow PKI," which can be used with standard digital signatures on an ongoing basis, makes possible a number of real-time applications that would otherwise have been infeasible.

In the naive implementation of an anonymous communication system described above, a new ring signature is made for every message sent within the system. Thus, if we say that m messages are sent in total among all n parties using the system, the total number of ring signatures that must be computed is

$O(m)$, and the number of ring signature verifications that take place is $O(nm)$, assuming every party receives every message.

On the other hand, our anonymization scheme allows parties to distribute keys in any signature algorithm and use them thereafter without giving away their identity. Thus, the number of ring signatures does not scale with the number of messages sent within the system. In particular, the number of ring signatures is only $O(n)$, one for each individual to sign their new key, the number of ring signature verifications is $O(n^3)$, (in the worst case, each of $O(n)$ Byzantine nodes can send different signatures to $O(n)$ honest nodes, resulting in a total of $O(n^2)$ verifications by each honest node), and the number of linking operations is $O(n^3)$ (with $O(n^2)$ messages to test for links, a node can maintain a list of pairwise-unlinked signatures and only test against those).

Thus, our scheme is ideally suited to application where a large number of messages are sent, relative to the number of participants. In the next section of this paper, the magnitude of the performance advantage provided by the Shadow PKI Protocol, as opposed to linkable ring signatures alone, will be demonstrated experimentally.

IV. Experimental Evaluation

A. Logistics

We used the Amazon Web Services (AWS) platform to construct several large-scale experimental distributed systems to evaluate the performance of proof-of-concept implementations of our anonymous consensus and Shadow PKI Protocols. Each system consisted of a variable number of t2.nano EC2 instances, each with 512MiB of DDR4 ECC RAM and a single 3.3 GHz vCPU core [33]. Ten such systems were constructed in total corresponding to network sizes of between 2 (2^1) and 1024 (2^{10}) instances (while "privacy" does not make much sense in a network of two parties, the performance data from the smallest network of $n = 2$ nodes is still potentially valuable). Each system was wholly located within a single availability zone (AZ) in the us-east-2 (Ohio) region, resulting in a mean observed network latency of 0.42 ms and bandwidth of 505 Mbps between any two instances over the local network. Our protocols were implemented in JavaScript and run using NodeJS 14.17.3 LTS on Ubuntu Server 20.04 LTS with uncompressed WebSockets for networking.

We ran a number of performance benchmarks on each distributed system to collect baseline metrics on the underlying networking and compute operations. We then ran multiple full executions of our reconstitution and consensus protocols on each system while measuring the execution time of every step of each protocol and testing the correctness of all intermediate and final values. Finally, we ran simulated application scenarios on each network. We chose to simulate a basic group messaging application where a closed group of authenticated participants could pseudonymously send messages to the entire group.

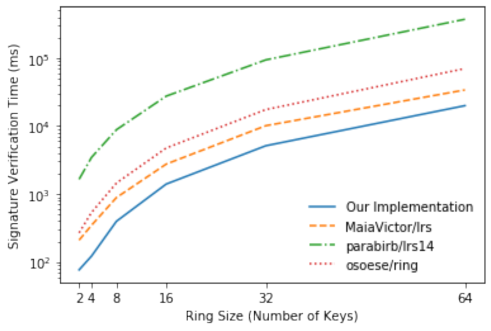

Fig. 3. Verification time for linkable ring signature schemes

B. Primitives

To maximize the versatility of our protocols, our specifications leave four choices of primative operations open to implementer preference: a digital signature scheme, a linkable ring signature scheme, a reliable broadcast protocol, and a covert broadcast protocol.

TABLE II
Verification time for digital signature schemes

Scheme	Time		
2048-bit RSA	5 ms	secp256k1	6 ms
4096-bit RSA	5 ms	p384	7 ms
8192-bit RSA	6 ms	brainpoolP384r1	7 ms
p256	6 ms	brainpoolP512r1	9 ms
brainpoolP256r1	6 ms	p521	11 ms
		curve25519	25 ms

1) Digital Signature Scheme: After evaluating 11 widely-available digital signature schemes, we decided to use 2048-bit RSA. We optimized for fastest verification time, noting that in both the PKI reconstitution and private chat application test cases, nodes are required to construct only a single signature but need to perform n signature verifications. The observed time (in ms) for a single signature verification in our environment for the 11 schemes we considered is shown in Table II.

2) Linkable Ring Signature Scheme: We considered several existing JavaScript linkable ring signature implementations but found that none of them were sufficiently performant for our use case, and therefore decided to proceed with our own implementation based on a JavaScript port of the Fujisaki traceable ring signature scheme [19]. The relative performance of our scheme versus existing schemes for various ring sizes is shown in Figure 3. While limited by fundamental truths about the asymptotic performance of linkable ring signatures, our implementation performs nearly 10x better than the others in absolute terms due to the use of Web Assembly rather than native JavaScript for heavy cryptographic operations.

3) Reliable Broadcast Protocol: We chose to use a gossip protocol for reliable broadcasting. While this approach reduces the consensus properties to only a probabilistic guarantee, the security for any given message remains extremely high with well-chosen network parameters [34] while still providing a significant performance improvement. In our tests, a network of 64 nodes took 8.9s to achieve all-to-all broadcast consensus on 128 1kB values using Dolev-Strong [6], while the same configuration took just 1.2s using a gossip protocol with a degree of 8, a 7.4x performance improvement, while still providing 99.6% probabilistic security against a 50% adversary.

4) Covert Broadcast Protocol: We chose to use the Dandelion protocol [23] for covert broadcasting. This was a natural fit for our network architecture as it behaves like an internal mix network during the "stem" phase while being nearly identical to our normal gossip broadcast method during the "fluff" phase. While hard to measure precisely, we estimate this method to be at least 100x more efficient than building a covert broadcast protocol on top of a point-to-point wide-area anonymity-preserving network like Tor [5].

C. Architecture

The use of a gossip protocol for reliable broadcast and the Dandelion protocol for covert broadcast requires careful selection of network parameters; namely, degree (neighbors per node) and Dandelion stem length. To avoid these parameters becoming confounding factors in our performance results, we chose to use a fixed stem size and gossip degree of 32 for all networks of at least 32 nodes, with this value providing the best balance of performance and security in our estimation. For networks smaller than 32 nodes, a complete (fully-connected) graph was used. The security characteristics for this architecture across various network sizes and various adversarial percentages are shown in Table III.

TABLE III
SECURITY PROBABILITIES OF CHOSEN NETWORK ARCHITECTURE

	Adversarial %						
N	**10%**	**25%**	**40%**	**50%**	**60%**	**75%**	**90%**
2	99.0%	93.8%	84.0%	75.0%	64.0%	43.8%	19.0%
4	100%	99.6%	97.4%	93.8%	87.0%	68.4%	34.4%
8	100%	100%	99.9%	99.6%	98.3%	90.0%	57.0%
16	100%	100%	100%	100%	100%	99.0%	81.5%
32	100%	100%	100%	100%	100%	100%	96.6%
64	100%	100%	100%	100%	100%	100%	96.6%
128	100%	100%	100%	100%	100%	100%	96.6%
256	100%	100%	100%	100%	100%	100%	96.6%
512	100%	100%	100%	100%	100%	100%	96.6%
1024	100%	100%	100%	100%	100%	100%	96.6%

In addition to the (up to 1,024) t2.nano "primary" instances, each network contained a single t2.medium "core" instance which played no active role in protocol execution and served solely as a bridge to connect each primary instance to 32 randomly chosen neighbors and to collect performance results after protocol termination. An offsite "remote" process communicated asynchronously with the core instance via HTTP to schedule experiments as well as to download, store, and aggregate the results.

D. Optimizations

Thus far, we have discussed a major optimization in the design of our protocol to minimize use of anonymous broadcasting. We have also discussed the optimal choice of digital signature scheme, linkable ring signature scheme, reliable broadcast protocol, and covert broadcast protocol for our setup. What follows is several more optimization considerations for the Anonymous Consensus Protocol implementation. Where relevant, we chose to optimize for the performance of a network with all non-faulty nodes as this was the scenario tested in our final demonstration.

1) Three-Phase Gossip: A significant potential bottleneck in our Anonymous Consensus Protocol is the agreement phase, in which every node rebroadcasts the set of inputs received in the input phase, requiring a network bandwidth utilization of $O(n^3)$. However, in a network with no faulty nodes, every node should receive the same set of messages in the anonymous input phase, and should therefore broadcast an identical message in the agreement phase. A three-phase gossip protocol reduces the bandwidth utilization of this stage to $O(n^2)$ by eliminating the need to broadcast redundant messages.

In the three-phase protocol, a node first gossips the hash of a message it intends to broadcast. If any node has not seen a message with that hash, it responds by requesting the full message. The first node then gossips the full message as normal. While increasing the latency of each message by at least threefold, this technique significantly reduces overall bandwidth utilization if many identical messages are expected, as is the case in our consensus stage. In our tests, this optimization reduced the time for 48 honest nodes to reach consensus on 48 random 1kB values from 112.8s to 9.7s, an 11.6x performance improvement. As such, we implemented this optimization in our final evaluation.

2) Reduced Linking: Another potential bottleneck is the need to perform n^2 link operations on the n received linkable ring signatures. There are a variety of optimizations one can make to the Anonymous Consensus Protocol to reduce the number of linking operations nodes need to perform. The first optimization relies on the observation that an honest node will always receive its own value in the input phase, and will thus always include its own input in the set it broadcasts during the agreement phase. If n parties all broadcast the same set in the agreement phase, it is guaranteed that the set contains at least n unlinked signatures. It is therefore safe to stop checking links when only n unique signatures remain. If only n unique signatures are present in the n broadcasted sets, as is the case when all n parties are non-faulty, there is no need to check signature links at all. In our tests, this optimization reduced the time for 32 honest nodes to reach consensus on 32 random values by 85%, a 6.7x performance improvement. There is no scenario in which this optimization would result in worse performance or security than the baseline. As such, we implemented this optimization in our final evaluation.

3) Optimized Linking: While checking for linked signatures, our baseline protocol specifies that signatures should not be removed until all links are checked (so that if many signatures are linked to the same party, they will all be detected). However, an important observation is that it is not necessary to keep more than one linked signature from any given party; when two signatures from the same party are found, one can immediately be discarded (and does not need to be checked against any other signatures).

Another observation is that in the sets of ring signatures that the protocols maintain over the course of the protocol, it is generally unnecessary for nodes to ever hold more than two linked signatures from any party in these sets at any time. This is due to the fact that two signatures which are linked to each other are sufficient to prove that the entity that made those signatures is malicious. Thus, if this set is later broadcast, it is not necessary to include a third or fourth signature which links to this same address. Nodes can avoid holding more than $2N$ signatures in a set by checking links for incoming set elements against each of the existing members in the set (though checking for links with both members of a linked pair is unnecessary) and discarding the element if it would be the third signature linked to an existing signature in the set.

While both of the above optimizations could provide performance improvements in the case where a small number of malicious parties attempt to include many values each, no performance improvement is realized for a network without Byzantine parties. We therefore did not include these optimizations in our evaluation.

4) Coordinated Linking: In the case where lots of links must be checked, a further performance benefit could be realized by distributing the workload across all of the honest nodes. A simple way to accomplish this would be for each node to check links starting from a random element in the set, and immediately broadcast information about a link if one has been discovered. This could theoretically reduce the number of links that need to be checked by any given node by $1/N$. Once again, because no performance improvement is realized for a network without malicious parties, we did not include this optimizations in our evaluation.

5) One-Time Ring Signatures: One-time Ring Signatures [29], [30] provide as a primitive a ring signature for which any key that produces a signature necessarily also produces a deterministic function of the private key. This provides a similar feature to linkable ring signatures in that any reuse of the key will allow the signature to be linked to any other signature with the same key.

The performance benefits of using one-time ring signatures over linkable ring signatures are twofold. The first is the reduction in linking time for n signatures from $O(n^2)$ to $O(n \log n)$ by the construction of an index of all key images. The performance gains are effectively even better than this due to the link operation itself being much simpler. This benefit is not realized in the network without Byzantine parties due to the Reduced Linking operation eliminating the need to check links entirely in that scenario. The second benefit is the ability to support a more efficient (non-linkable) underlying ring signature scheme. According to some estimates [31], the use of Arcturus [32] ring signatures could reduce verification times by as much as 96% from our current scheme (Fujisaki-Suzuki [19]).

Although this performance improvement is potentially very promising, we decided against implementing it for a few reasons. The simplest reason is the lack of any efficient JavaScript implementation of one-time ring signatures and the difficulty of producing such an implementation from scratch. Another reason is the difficulty of recovering from consensus failures without having to repeat the entire reconstitution phase; in the Fujisaki-Suzuki scheme, by contrast, this is as simple as choosing another ‘issue’ value and trying again. Lastly, it is not clear whether one-time ring signatures are compatible with our proofs for the consensus security and privacy properties, whereas the oracle construction of the Fujisaki-Suzuki scheme can be used to produce relatively straightforward proofs (see Appendix D-A).

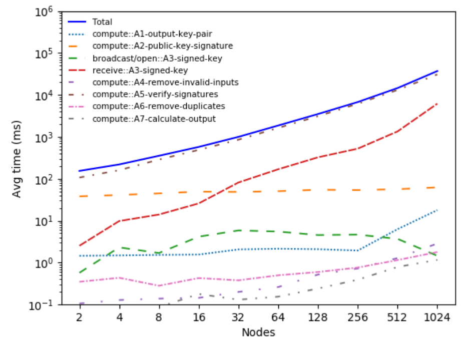

Fig. 4. Stepwise performance of PKI Reconstitution Protocol

E. Results

1) PKI Reconstitution: We begin by discussing the results of our evaluation of the PKI Reconstitution Protocol. Reconstituting a PKI from a standard digital signature scheme to a linkable ring signature scheme serves as an important precursor to the Anonymous Consensus and Shadow PKI Protocols as well as both of our tested application scenarios. The time for n processes to achieve reconstitution from a 2048-bit RSA digital signature scheme to a Fujisaki-Suzuki traceable ring signature scheme varied from 0.15s for $n = 2$ to 37.1s for $n = 1024$ according to a roughly linear relationship.

The PKI Reconstitution Protocol is divided into 7 major stages, with the broadcast stage being further divided into broadcast and receive stages. The time taken to execute each stage of the protocol, along with the total execution time, is shown in Figure 4. It is clear that the signature verification stage (Stage A5) is by far the largest factor in the overall performance of the PKI Reconstitution Protocol; for example, when $n = 1024$, 30.8 seconds of the 37.1 second execution time are spent verifying signatures, while just 6.3 seconds are spent on all of the remaining 6 stages of the protocol combined. Since each node must verify n signatures, and signature verification is the dominant performance term, we would expect the overall time complexity of the PKI Reconstitution Protocol to be $O(n)$, which matches our observations.

Although our selection of 2048-bit RSA for signatures was intended to minimize signature verification time, further optimizations may exist in light of the ability to verify signatures in batches. For instance, batch verification of EdDSA signa-

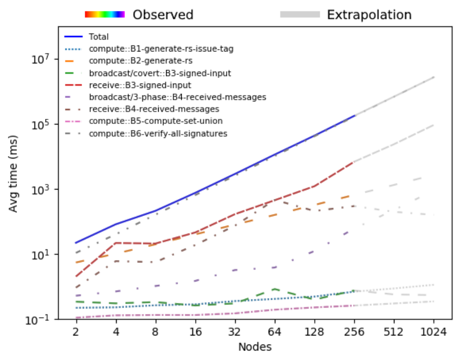

Fig. 5. Stepwise performance of Anonymous Consensus Protocol

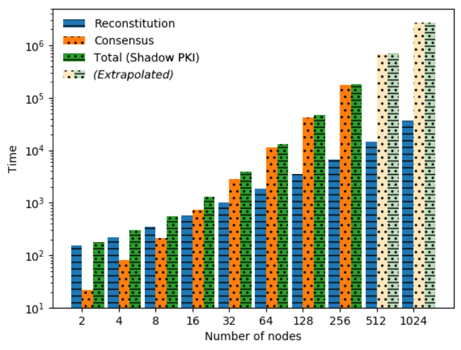

Fig. 6. Stepwise performance of Shadow PKI Protocol

tures is claimed to achieve up to 71,000 signature verifications per second [35]. If the full benefit of this approach could be realized, it might theoretically reduce the reconstitution time for 1024 nodes from 37.1s to 6.4s, a 5.8x improvement. Still, it's worth emphasizing that PKI reconstitution is almost always a single-use operation; after a single reconstitution into a suitable linkable ring signature scheme, the reconstituted PKI could be used for an unlimited number of subsequent consensus iterations without requiring further reconstitution. Thus, given the difficulty of coordinating communications in a large distributed system and the one-time-use nature of the PKI Reconstitution Protocol, we consider an end-to-end PKI reconstitution time of under 1 minute for a huge system of 1,024 nodes to be an acceptable result.

2) Anonymous Consensus: Our evaluation of the Anonymous Consensus Protocol begins in the state following execution of the PKI Reconstitution Protocol in the previous experiment; namely, with a PKI fully reconstituted into a traceable ring signature scheme. We then measure the time for n processes to achieve consensus on n anonymous inputs (one from each process). The consensus protocol took between 22 ms for $n = 2$ and 174s for $n = 256$ according to a roughly quadratic relationship.

The Anonymous Consensus Protocol is divided into 6 major stages, with the two broadcast stages (B3, covert/input and B4, open/consensus) being further divided into broadcast and receive stages. The time taken to execute each stage of the protocol, along with the total execution time, is shown in Figure 5. It is clear that the ring signature verification stage (Stage B6) is by far the largest factor in overall performance; for example, when $n = 256$, 166 seconds of the 174 second execution time are spent verifying signatures, while just 8 seconds are spent on all of the other protocol stages. Since each node must verify n ring signatures in this stage, and ring signature verification is typically $O(n)$ with respect to the ring size, we would expect the overall time complexity of the Anonymous Consensus Protocol to be $O(n^2)$, which matches our observations.

Due to time and resource constraints, execution of the anonymous consensus experiments was limited to 10 minutes per system. Unfortunately, the networks of 512 and 1024 nodes did not finish executing in the allotted time. We have extrapolated the results from the remaining 8 networks, for which exact data was available, to produce power-law estimations for the networks of 512 and 1024 nodes. The portion of Figure 5 which refers to extrapolated data is shown in grey. This extrapolation suggests that 1024 nodes would have taken about 44 minutes to reach consensus on 1024 anonymous inputs, with 99% of this time being spent verifying signatures.

The computational difficulty of verifying a large number of linkable ring signatures with large ring sizes makes our current setup potentially infeasable for distributed systems with more than 256 nodes (taking 174 seconds). However, in the next sections, we will demonstrate that for many applications, linkable ring signatures are themselves the only feasible alternative to the use of a Shadow PKI Protocol via the Anonymous Consensus algorithm. Thus, the one-time deployment of a Shadow PKI Protocol using our anonymous consensus technique remains more efficient than ring signatures alone, and its relative efficiency is, perhaps counterintuitively, actually *amplified* for large networks rather than the effect being diminished.

Nevertheless, more efficiently verifiable linkable ring signature schemes could have an outsized impact on the scalability of our Anonymous Consensus Protocol. Our future work section describes a number of potential approaches for improving performance in this respect. If these approaches allowed $O(n)$ verification of linkable ring signatures, performance of the anonymous consensus on 1024 nodes could be improved by over 2.5x even if each signature took a full second to verify. We therefore consider this a primary area of focus for future study.

3) Shadow PKI: The PKI Reconstitution Protocol and Anonymous Consensus Protocol are the components of our Shadow PKI (PKI pseudonymization) protocol, which consists fundamentally of applying the two protocols in series. The final component of the Shadow PKI Protocol is simply generating a key pair; however, because this is a fixed-cost, non-interactive computation, it can be pre-computed or computed in parallel with network stages (e.g. A3, B3, B4) without impacting total execution time. We therefore for performance evaluation consider the Shadow PKI Protocol to be a simple sequential execution of the PKI Reconstitution and Anony-

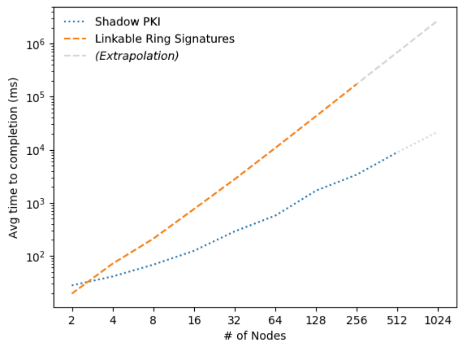

Fig. 7. Linkable Signature vs. Shadow PKI for a Private Chat Message

mous Consensus Protocol.

Figure 6 shows the PKI Reconstitution, Anonymous Consensus, and PKI pseudonymization (Reconstitution + Consensus) times for each system. Once again, a few of the displayed values are extrapolations, and these are displayed in lighter color on the figure. We observe that reconstitution dominates the Shadow PKI performance for smaller networks, representing 87% of the total execution time for $n = 2$. Consensus overtakes reconstitution after $n = 16$, reaching 99% of total execution time for $n = 1024$. We believe this is due to the fact that although linkable ring signatures have worse asymptotic performance than normal ring signatures, they typically have better absolute performance for small ring sizes due to the use of much smaller security parameters. The total time to execute the Shadow PKI algorithm varied from 176ms for $n = 2$ to an observed maximum of 181s for $n = 256$ or an estimated 45 minutes for $n = 1024$, with ring signature verification remaining the predominant bottleneck for larger networks.

4) Application Scenario: To understand the performance impact of the Shadow PKI Protocol on realistic practical application scenarios, we designed a simulation of a simple pseudonymous group messaging application, whereby members are admitted to a chat group based on a key linked to their identity, and can then post chat messages pseudonymously to the entire group (such that two messages from the same user are still linked to eachother). This application can be accomplished using the Shadow PKI technique or with linkable ring signatures alone, with both methods providing identical functionality.

We constructed a basic simulation of the application using both techniques, with every user in the group sending exactly one message. The goal of these simulations are to understand the ongoing performance benefits of using the Shadow PKI scheme over ring signatures alone, so that it can be determined how quickly the initial computational cost of the Shadow PKI Protocol would be recouped by its long-term relative performance advantage. As such, we assume that the initial PKI transformations have already occured when running the simulations. For the Shadow PKI version of the application, we assume that the Shadow PKI Protocol has already been executed and simply measure the time for every user to sign and anonymously (Dandelion) broadcast a single message using the Shadow PKI (2048-bit RSA) and to verify the messages of all other users. Similarly, for the Linkable Ring Signatures version of the application, we assume that the PKI Reconstitution Protocol has already been executed and a PKI with respect to a Linkable Ring Signature scheme is available. We then measure the time for every user to sign and anonymously (Dandelion) broadcast a single message using a linkable ring signature and to verify the messages of all other users. The results of this simulation are shown in Figure 7. Once again, some portions were extrapolated due to time constraints, shown in lighter color.

As expected, the Shadow PKI approach is significantly more efficient for sending a single message than ring signatures alone for all networks greater than size $n = 2$ (the reason for this anomaly is the same as the one discussed in the previous section). For $n = 256$, corresponding to the chat group of 256 users each sending a single message, the Shadow PKI provided over 51 times better performance than ring signatures. Extrapolating this to 1024 users, the Shadow PKI could provide over 126 times better performance. While the relative advantage of using normal signatures with a Shadow PKI, so too does the initial cost of executing the Shadow PKI (PKI pseudonymization) protocol. What remains is to determine the relative efficiency of each approach, considering the upfront cost of consensus and pseudonymization along side the long-term benefit to determine which scenarios particularly advantage the Shadow PKI technique.

Table IV, which shows how many times more efficient the Shadow PKI approach is than linkable ring signatures alone for a network of n users sending m messages each, now including the upfront costs of each approach (PKI transformation) in addition to the ongoing costs (generating and verifying signatures). For example, in the case of 256 users ending 128 messages each, we have determined that the Shadow PKI method performs over 36x better than linkable ring signatures.

TABLE IV
EFFICIENCY IMPROVEMENT PROVIDED BY SHADOW PKI

	Messages per Node										
N	**1**	**2**	**4**	**8**	**16**	**32**	**64**	**128**	**256**	**512**	**1024**
2	0.85	0.83	0.81	0.78	0.75	0.73	0.72	0.71	0.70	0.70	0.70
4	0.85	0.95	1.08	1.26	1.42	1.55	1.64	1.69	1.71	1.73	1.73
8	0.89	1.10	1.43	1.83	2.25	2.58	2.80	2.94	3.01	3.05	3.07
16	0.92	1.33	1.98	2.87	3.85	4.70	5.31	5.69	5.90	6.01	6.07
32	0.91	1.47	2.41	3.76	5.36	6.86	8.00	8.74	9.16	9.38	9.50
64	0.93	1.65	2.93	5.00	7.86	11.07	13.96	16.06	17.36	18.10	18.49
128	0.97	1.80	3.30	5.79	9.40	13.72	17.81	20.95	22.98	24.14	24.77
256	0.98	1.88	3.60	6.70	11.83	19.20	27.90	36.09	42.30	46.29	48.58
512	1.01	1.97	3.82	7.26	13.26	22.61	34.93	48.04	59.13	66.85	71.51
1024	1.03	2.03	3.99	7.71	14.53	26.06	43.20	64.39	85.32	101.88	112.82

As shown in Table IV, the shadow PKI approach is more efficient than ring signatures alone for the group chat scenario in any network with 4 or more participants sending 4 or more messages each. At the extreme range of our tests, groups of 1024 participants sending 512 or more messages each experience over 100x better performance using the PKI pseudonymization technique compared to using ring signatures on every message. We have no reason to believe this trend wouldn't continue for even larger hypothetical networks. We therefore consider the Shadow PKI approach highly successful from a performance perspective, despite the seemingly large upfront computational cost in absolute terms. Improvements to the Anonymous Consensus Protocol brought by advancements in linkable ring signature efficiency would serve only to further magnify this performance advantage.

V. Conclusion

In this paper, we have demonstrated a collection of protocols for the purpose of distributing a Shadow PKI. This includes a basic PKI Reconstitution Protocol for switching between cryptosystems, as well as an anonymous multi-value all-To-all broadcast protocol for distributing keys or other values anonymously.

We have furthermore provided a description of the properties that we need from ring signatures and network primitives to make these protocols function securely. We have provided intuition for the behavior of these protocols within the text and provided rigor in the appendix.

Our experimental results have demonstrated that these protocols have applications in a number of useful scenarios, in particular when the number of many-to-many messages that anonymous system needs to support is large. The results that have come out of our demonstration framework and the code we have created to implement all of the proposed algorithms are open source and publicly available for your consideration.

A. Future Work

Due to the modularity of the protocols we have presented, we think there are fruitful directions for future work in terms of examining different choices for instantiation of our protocols is worthwhile to ask in what ways they could be instantiated.

One collection of ideas worth exploring are those from the optimizations section. In particular, the linking operation can be expensive for ring signatures, so ways of avoiding it (by somehow distributing the checking of links, or adopting one time ring signatures) could be beneficial.

Another potential optimization along these lines is in the choice of cryptography behind the ring signature itself. In particular, rather than the Diffie-Hellman based Fujisaki-Suzuki ring signature presented here, one could instead use a ring signature based on zk-SNARKs [12]. This would provide anonymization of keys in a way functionally similar to the blinding system of Zcash [36]. Constructions using this approach would have various downside and upsides in terms of the strength of their cryptographic assumptions, as well as their compute and communication costs, depending on the zk-SNARKs used. While we have sidestepped these issues by focusing on simpler cryptographic primitives, zk-SNARKs offer the opportunity of constant size signatures for less network and verifier overhead, and they deserve consideration.

We also wish to examine the applicability of our Shadow PKI protocol to secure multiparty computation. While techniques for anonymizing MPC protocols have been proposed, they involve significant computational overhead that could potentially be minimized by operating the entire MPC scheme under a Shadow PKI.

Another dimension for examination is in the broadcast protocols used in the consensus steps of our algorithms. In particular, our constructions have the nice property that they inherit the security properties of the protocols they are instantiated with. In a setting with a large number of participants, a protocol with strong security guarantees like Dolev-Strong [6] might be compute intensive, but would result in a very robust protocol. Future work could examine better in which situations this trade-off and other like it are worth making.

Acknowledgment

The authors would like to thank the various individuals who provided feedback on this project, as well as funding agencies that provided support. Full details to be released in the published form of this manuscript.

References

[1] Blum, Manuel and Feldman, Paul and Micali, Silvio, *Non-Interactive Zero-Knowledge and its Applications*, Proceedings of the twentieth annual ACM symposium on Theory of computing - STOC 88, 1988.

[2] Castro, Miguel and Liskov, Barbara, *Practical Byzantine Fault Tolerance and Proactive Recovery*, ACM Transactions on Computer Systems, 2002.

[3] Choi, Byeong-Cheol and Park, Sohee and Kim, Jungnyea and Ryou, Jeicheoul, *Anonymous PKI Framework for Privacy-Guaranteed e-Services*, 2007 International Conference on Convergence Information Technology (ICCIT 2007), 2007.

[4] Critchlow, D. and Zhang, N., *Security Enhanced Accountable Anonymous PKI Certificates for Mobile e-Commerce*, Computer Networks, 2004.

[5] Dingledine, Roger and Mathewson, Nick and Syverson, Paul, *Tor: The Second-Generation Onion Router*, USENIX Security Symposium, 2004.

[6] Dolev, D. and Strong, H. R., *Authenticated Algorithms for Byzantine Agreement*, SIAM Journal on Computing, 1983.

[7] Groth, Jens, *On the Size of Pairing-Based Non-interactive Arguments*, Advances in Cryptology – EUROCRYPT 2016 Lecture Notes in Computer Science, 2016.

[8] Hendrickx, Julien M. and Tsitsiklis, John N., *Fundamental Limitations for Anonymous Distributed Systems with Broadcast Communications*, 2015 53rd Annual Allerton Conference on Communication, Control, and Computing (Allerton), 2015.

[9] Lamport, Leslie, *The Part-Time Parliament*, ACM Transactions on Computer Systems, 1998.

[10] Magnani, Antonio and Dangelo, Gabriele and Ferretti, Stefano and Marzolla, Moreno, *Anonymity and Confidentiality in Secure Distributed Simulation*, 2018 IEEE/ACM 22nd International Symposium on Distributed Simulation and Real Time Applications (DS-RT), 2018.

[11] Mccoy, Damon and Bauer, Kevin and Grunwald, Dirk and Kohno, Tadayoshi and Sicker, Douglas, *Shining Light in Dark Places: Understanding the Tor Network*, Privacy Enhancing Technologies Lecture Notes in Computer Science, 2008.

[12] Parno, B. and Howell, J. and Gentry, C. and Raykova, M., *Pinocchio: Nearly Practical Verifiable Computation*, 2013 IEEE Symposium on Security and Privacy, 2013.

[13] Rivest, Ronald L. and Shamir, Adi and Tauman, Yael, *How to Leak a Secret*, Advances in Cryptology — ASIACRYPT 2001 Lecture Notes in Computer Science, 2001.

[14] Wang, Chuan-Jun and Niu, Xia-Mu and Zhang, Yong, *Anonymity in PKI Environment*, Third International Conference on Intelligent Information Hiding and Multimedia Signal Processing (IIH-MSP 2007), 2007.

[15] Benaloh, Josh, *Simple Verifiable Elections*, Proceedings of the USENIX/Accurate Electronic Voting Technology Workshop 2006 on Electronic Voting Technology Workshop, 2006.

[16] Nicolosi, Antonio R., *Authentication Mechanisms for Open Distributed Systems*, New York University, 2007.

[17] Tang, Jian and Larrea, Mikel and Arevalo, Sergio and Jimenez, Ernesto, *Implementing Uniform Reliable Broadcast in Anonymous Distributed Systems with Fair Lossy Channels*, 2015 IEEE International Parallel and Distributed Processing Symposium Workshop, 2015.

[18] Ciaraldi, Michael and Finkel, David and Wills, Craig, *Risks in anonymous distributed computing systems*, International Network Conference 2000, 2000.

[19] Fujisaki, Eiichiro and Suzuki, Koutarou, *Traceable Ring Signature*, Public Key Cryptography – PKC 2007 Lecture Notes in Computer Science, 2007.

[20] Tsang, Patrick P. and Wei, Victor K., *Short Linkable Ring Signatures for E-Voting, E-Cash and Attestation*, Information Security Practice and Experience Lecture Notes in Computer Science, 2005.

[21] Camenisch, Jan and Lysyanskaya, Anna, *An Efficient System for Non-transferable Anonymous Credentials with Optional Anonymity Revocation*, Lecture Notes in Computer Science Advances in Cryptology — EUROCRYPT 2001, 2001.

[22] Garman, Christina and Green, Matthew and Miers, Ian, *Decentralized Anonymous Credentials*, Proceedings 2014 Network and Distributed System Security Symposium, 2014.

[23] Giulia C. Fanti, Shaileshh Bojja Venkatakrishnan, Surya Bakshi, Bradley Denby, Shruti Bhargava, Andrew Miller and Pramod Viswanath, *Dandelion++: Lightweight Cryptocurrency Networking with Formal Anonymity Guarantees*, CoRR, 2018.

[24] Chaum, David L., *Untraceable Electronic Mail, Return Addresses, and Digital Pseudonyms*, Commun. ACM, 1981.

[25] Tim Ruffing, Pedro Moreno-Sanchez, Aniket Kate, *P2P Mixing and Unlinkable Bitcoin Transactions*, NDSS Symposium, 2017.

[26] James Aspnes, Collin Jackson, Arvind Krishnamurthy, *Exposing Computationally-Challenged Byzantine Impostors*, Technical Report YALEUDCSTR-1332, 2004.

[27] Haenni, Rolf and Spycher, Oliver, *Secure Internet Voting on Limited Devices with Anonymized DSA Public Keys*, EVT/WOTE, 2011.

[28] Stefan Kaufmann, *Implementation and Adaptation of the Pseudonymous PKI for Ubiquitous Computing for Car-2-Car Communication*, Automotive Safety and Security, 2014.

[29] Nicolas van Saberhagen, *CryptoNote v 2.0*, Bytecoin.org, 2013.

[30] Wang X., Chen Y., Ma X., *Adding Linkability to Ring Signatures with One-Time Signatures*, Information Security, 2019.

[31] IBurnMyCD, *Ring Signature Performance Metrics*, TurtleCoin, 2021.

[32] Sarang Noether, *Arcturus: Efficient Proofs for Confidential Transactions*, Monero Research Lab, 2020.

[33] Amazon Web Services Inc., *Amazon EC2 Instance Types*, `https://aws.amazon.com/ec2/instance-types/`, 2020.

[34] Demers, Alan; Greene, Dan; Hauser, Carl; Irish, Wes; Larson, John; Shenker, Scott; Sturgis, Howard; Swinehart, Dan; Terry, Doug, *Epidemic Algorithms for Replicated Database Maintenance*, Association for Computing Machinery, 1987.

[35] Bernstein, D.J., Duif, N., Lange, T. et al., *High-speed High-security Signatures.*, J Cryptogr Eng, 2012.

[36] Banerjee, Aritra, Michael Clear, and Hitesh Tewari. *Demystifying the Role of zk-SNARKs in Zcash.* IEEE Conference on Application, Information and Network Security (AINS). 2020.

Appendix A
Formalization

A. Public Key Enablement

Definition 1. *A* public key cryptography system

$$\Sigma = (\mathsf{Gen}, \mathsf{Sig}, \mathsf{Ver}[, \mathsf{Link}][, \mathsf{Trace}][, Enc, Dec][, \ldots])$$

is a tuple of algorithms minimally enabling digital signatures (and optionally enabling additional functionality). We expect Gen, Sig, *and* Ver *to nominally behave according to definition 135.1 of Pass and Shelat.*

Remark 1. *Asymmetric encryption (*Enc *and* Dec*) is not used in any of the protocols presented in this paper, but is required for many practical applications of public key cryptography.*

Definition 2. *A* linkable ring signature scheme

$$L = (\mathsf{Gen}, \mathsf{Sig}, \mathsf{Ver}, \mathsf{Link})$$

is a public key cryptography scheme satisfying unforgeability, L-anonymity, and linkability, nominally according to the definitions of Tsang and Wei.

Public key cryptography takes place in the context of a public key enablement containing identities. A correspondence exists between identities and public keys. Let $\mathrm{ID}(\mathrm{pk}) = \mathrm{id}$ denote the identity associated with public key pk (i.e., the identity knowing the secret key sk corresponding to the public key pk), and $\mathrm{PKS}(\mathrm{id}) = \{\mathrm{pk}_1, \mathrm{pk}_2, \ldots \mathrm{pk}_n\}$ denote the set of public keys associated with identity id, whereby $\forall\, \mathrm{pk} \in \mathrm{PKS}(\mathrm{id}), \mathrm{ID}(\mathrm{pk}) = \mathrm{id}$.

Definition 3. *An* identity partition

$$P_{\langle\Sigma\rangle} = (\{\mathrm{pk}_1, \mathrm{pk}_2, \ldots \mathrm{pk}_n\}, \{\mathrm{sk}_1, \mathrm{sk}_2, \ldots \mathrm{sk}_n\})$$

with respect to a public key cryptography scheme Σ is a pair containing a set of identities and a set of public keys in the output space of the Gen *algorithm of Σ: $pk_i \in \{\mathrm{pk}\,|(\mathrm{pk}, \mathrm{sk}) \in \Sigma.\, \mathsf{Gen}^{\rightarrow}\}$, each of the public keys belonging to an identity in $P_{\langle\Sigma\rangle}$: $\forall\, \mathrm{pk} \in P_{\langle\Sigma\rangle}, \mathrm{ID}(\mathrm{pk}) \in P_{\langle\Sigma\rangle}$.*

Property 1. *An identity partition $P_{\langle\Sigma\rangle}$ is* Sybil resistant *if at most one public key is associated with each identity in $P_{\langle\Sigma\rangle}$: $\forall id \in P_{\langle\Sigma\rangle}, |\, \mathrm{PKS}(\mathrm{id})| \leq 1$.*

Remark 2. *If an identity partition*

$$P_{\langle\Sigma\rangle} = (\{\mathrm{pk}_1, \mathrm{pk}_2, \ldots \mathrm{pk}_n\}, \{\mathrm{sk}_1, \mathrm{sk}_2, \ldots \mathrm{sk}_n\})$$

is Sybil resistant the number of identities in $P_{\langle\Sigma\rangle}$ is at least as large as the number of public keys in $P_{\langle\Sigma\rangle}$: $m \geq n$.

Property 2. *An identity partition $P_{\langle\Sigma\rangle}$ is* anonymous *if a PPT adversary knowing $\mathrm{ID}(\mathrm{pk}_1), ID(\mathrm{pk}_2), \ldots ID(\mathrm{pk}_f)$ for $f < n$ public keys in $P_{\langle\Sigma\rangle}$ and having query access to a message oracle containing all prior public broadcasts will not be able to guess $ID(pk)$ for any other public key $\mathrm{pk} \in \mathrm{pk}_{f+1}, pk_{f+2}, \ldots \mathrm{pk}_n$ with greater than $1/(n-f) + \mathrm{negl}$ probability.*

Definition 4. *A* partitioned public key infrastructure $\mathrm{PPKI}_{\langle\Sigma\rangle}$ *is Sybil resistant if each of its identity partitions are Sybil resistant and identities exist in at most one partition: $\forall P_{\langle\Sigma\rangle,a}, P_{\langle\Sigma\rangle,b} \in \mathrm{PPKI}_{\langle\Sigma\rangle}\; s.t. a \neq b, \forall id_a \in P_{\langle\Sigma\rangle,a}, id_b \in P_{\langle\Sigma\rangle,b}, id_a \neq id_b$.*

Definition 5. *An* onymous public key infrastructure

$$\begin{aligned}&PPKI(O)_{\langle\Sigma\rangle}\\ &\quad = \{(\{id_1\}, \{\mathrm{pk}_1\}), (\{id_2\}, \{\mathrm{pk}_2\}), \ldots (\{id_n\}, \{\mathrm{pk}_n\})\}\end{aligned}$$

is a partitioned public key infrastructure where every partition $P_{\langle\Sigma\rangle,i}$ in $PPKI(O)_{\langle\Sigma\rangle}$ contains a single identity and corresponding public key: $\forall P_{\langle\Sigma\rangle} \in PPKI(O)_{\langle\Sigma\rangle}, P_{\langle\Sigma\rangle} = (\{ID(pk)\}, \{pk\})$.

Remark 3. *An onymous public key infrastructure that is Sybil-resistant mirrors the conventional understanding of a PKI — there is a one-to-one correspondence between identities and public keys with the number of partitions being equal to the number of identities in the PKI — thus, the identity corresponding to a given public key (and vice-versa) is obvious given $PPKI(O)_{\langle\Sigma\rangle}$.*

Definition 6. *A* semi-anonymous public key infrastructure

$$PPKI(S)_{\langle\Sigma\rangle} = \{P_{\langle\Sigma\rangle,1}, P_{\langle\Sigma\rangle,2}, \ldots P_{\langle\Sigma\rangle,n}\}$$

is a partitioned public key infrastructure with multiple partitions: $|PPKI(S)_{\langle\Sigma\rangle}| \geq 2$, and at least one partition with multiple identities: $\exists P_{\langle\Sigma\rangle} \in PPKI(S)_{\langle\Sigma\rangle}, |idsP_{\langle\Sigma\rangle}| \geq 2$, whereby each partition in $PPKI(S)_{\langle\Sigma\rangle}$ is anonymous.

Remark 4. *Given a semi-anonymous public key infrastructure $PPKI(S)_{\langle\Sigma\rangle}$, one can easily identify the partition that a public key belongs to, but not which identity within that partition it corresponds to.*

We call this PKI "semi-anonymous" because keys within the PKI are identifiable to a subset of the identities in the PKI. For instance, if partitions in a semi-anonymous PKI represent institutions and identities represent members of those institutions, one might be able to identify the institution to which a public key belongs but not the individual member.

Definition 7. *An* anonymous public key infrastructure

$$PPKI(A)_{\langle\Sigma\rangle} = \{(\{id_1, id_2, \ldots id_n\}, \{\mathrm{pk}_1, pk_2, \ldots \mathrm{pk}_n\})\}$$

is a partitioned public key infrastructure with a single partition $|PPKI(A)_{\langle\Sigma\rangle}| = 1$, and multiple identities in that partition: $PPKI(A)_{\langle\Sigma\rangle} = \{P_{\langle\Sigma\rangle}\}, |ids \in P_{\langle\Sigma\rangle}| \geq 2$, whereby the partition $P_{\langle\Sigma\rangle}$ in $PPKI(A)_{\langle\Sigma\rangle}$ is anonymous.

Remark 5. *Given an anonymous public key infrastructure $PPKI(A)_{\langle\Sigma\rangle}$, one cannot easily identity $ID(pk)$ given a random $pk \in PPKI(A)_{\langle\Sigma\rangle}$ beyond the knowledge that $ID(pk) \in PPKI(A)_{\langle\Sigma\rangle}$.*

B. PKI Transformation

Definition 8. *In the following protocols, let* process *refer to a node in a computer network belonging to an identity id where $id = ID(pk)$ for some pk in the partitioned PKI* PPKI_{IN}*. Let $P(id) = p$ denote the process belonging to id. Processes in a partitioned PKI* $\mathrm{PPKI}_{\langle\Sigma\rangle}$ *with respect to Σ are assumed to have access to the algorithms within Σ.*

Definition 9. *A* PKI transformation protocol *allows processes in a partitioned PKI* PPKI_{IN} *to achieve consensus on a second partitioned PKI* PPKI_{OUT}*.*

Property 3. *A PKI transformation protocol is* decentralized *if each non-faulty process executes an identical algorithm and no single non-faulty process or group of non-faulty processes has a special role or function distinct from the roles and functions of the other non-faulty processes.*

Property 4. *A PKI transformation protocol is* synchronous *if upon initiating the protocol at time t one can determine the time t' by which the protocol has terminated at all non-faulty processes.*

Property 5. *A PKI transformation protocol is* live *if all non-faulty processes terminate and output a partitioned PKI* PPKI_{OUT} *upon termination.*

Property 6. *A PKI transformation protocol is* safe *if the partitioned PKI* PPKI_{OUT} *output by any non-faulty processes is the same as the partitioned PKI* PPKI_{OUT} *output by all other non-faulty processes.*

Property 7. *A PKI transformation protocol is* Sybil-resistant *if* PPKI_{IN} *being Sybil-resistant implies that the partitioned PKI output by any non-faulty processes* PPKI_{OUT} *will also be Sybil-resistant.*

Property 8. *A PKI transformation protocol is* integrous *if all identities in* PPKI_{OUT} *are present in* PPKI_{IN}*: $id \in \mathrm{PPKI}_{OUT} \implies id \in \mathrm{PPKI}_{IN}$.*

Property 9. *A PKI transformation protocol is* inclusive *if all identities in* PPKI_{IN} *controlled by non-faulty processes are present in* PPKI_{OUT}*: $id \in \mathrm{PPKI}_{IN} \wedge P(id)$ is not faulty $\implies id \in \mathrm{PPKI}_{OUT}$.*

Definition 10. *A* complete *PKI transformation protocol satisfies the properties of safety, liveness, validity, Sybil-resistance, integrity, inclusivity, synchrony, and decentralization.*

Definition 11. *A* PKI Reconstitution Protocol *is a PKI transformation protocol that allows processes in a partitioned PKI* $\mathrm{PPKI}_{IN_{\langle\Sigma_1\rangle}}$ *with respect to public key cryptography scheme Σ_1 to achieve consensus on a second partitioned* $\mathrm{PPKI}_{OUT_{\langle\Sigma_2\rangle}}$ *with respect to a second public key cryptography scheme Σ_2.*

Property 10. *A PKI Reconstitution Protocol is* valid *if all processes output a valid partitioned public key infrastructure* $\mathrm{PPKI}_{OUT_{\langle\Sigma_2\rangle}}$ *with respect to Σ_2.*

Definition 12. *A* Shadow PKI Protocol *is a PKI transformation protocol that allows processes in a partitioned PKI* $\mathrm{PPKI}_{IN_{\langle\Sigma\rangle}}$ *to achieve consensus on an anonymous PKI $PPKI(A)_{OUT_{\langle\Sigma\rangle}}$.*

Property 11. *A Shadow PKI Protocol is* valid *if all processes output a valid anonymous public key infrastructure $PPKI(A)_{OUT_{\langle\Sigma\rangle}}$.*

Appendix B
Protocols

A. Requirements

Definition 13. *A broadcast protocol allows a message m broadcast by a sender to be received by all non-faulty processes.*

Property 12. *A broadcast protocol is* live *if all non-faulty processes output a value.*

Property 13. *A broadcast protocol is* safe *if no non-faulty processes output different values.*

Property 14. *A broadcast protocol is* valid *if all non-faulty processes output the sender's value if the sender is non-faulty.*

Property 15. *A broadcast protocol is* synchronous *if upon initiating the protocol at time t one can determine the time t' by which the protocol has terminated at all non-faulty processes.*

Property 16. *A broadcast protocol is* decentralized *if each non-faulty process executes an identical algorithm and no single non-faulty process or group of non-faulty processes has a special role or function distinct from the roles and functions of the other non-faulty processes.*

Definition 14. *A* reliable broadcast protocol *is a broadcast protocol satisfying the properties of safety, liveness, validity, synchrony, and decentralization.*

Remark 6. *An example of a reliable broadcast protocol in an authenticated setting with synchronous unicasts is Dolev-Strong.*

Property 17. *A broadcast protocol is* anonymous *if a PPT adversary knowing the identity of senders for messages sent by $f < n$ processes and having query access to a message oracle containing all prior public broadcasts will not be able to identify the sender of a message with greater than $1/(n-f) + \text{negl.}$ probability.*

Definition 15. *A* covert broadcast protocol *is a broadcast protocol satisfying the properties of synchrony, decentralization, and anonymity.*

Remark 7. *A covert broadcast protocol can be constructed in any authenticated setting with synchronous unicasts. Providing this construction may be the focus of future work. For the sake of this paper, covert broadcasts can also be constructed via a series of anonymous unicasts over the Tor network [5].*

B. PKI Reconstitution

1) Protocol: A complete PKI Reconstitution Protocol given a reliable broadcast protocol.

Algorithm 1 PKI Reconstitution Algorithm

```
$(\mathrm{pk}_{OUT,i}, \mathrm{sk}_{OUT,i}) \leftarrow \Sigma_2.\mathsf{Gen}$
$\sigma \leftarrow \Sigma_1.\mathsf{Sig}_{\mathrm{sk}_{IN,i}}(\mathrm{pk}_{OUT,i})$
$\mathrm{BroadcastSend}_i((\mathrm{pk}_{IN,i}, \mathrm{pk}_{OUT,i}, \sigma_i))$
for all $1 \le j \le n$ do
    $m_j \leftarrow \mathrm{BroadcastReceive}_j$
    $M \leftarrow M \cup \{m_j\}$
end for
for all $1 \le j \le n$ do
    if $\mathrm{pk}_{IN,j} \notin \mathrm{PPKI}_{IN_{\langle \Sigma_1 \rangle}}$ then
        $M \leftarrow M \setminus \{m_j\}$
    end if
end for
for all $1 \le j \le n$ do
    if $\Sigma_1.\mathsf{Ver}_{\mathrm{pk}_{IN,j}}(\mathrm{pk}_{OUT,j}, \sigma_j) = 0$ then
        $M \leftarrow M \setminus \{m_j\}$
    end if
end for
for all $1 \le a \le n$ do
    for all $1 \le b < a$ do
        if $pk_{IN,a} = pk_{IN,b}$ then
            $R \leftarrow R \cup \{m_a, m_b\}$
        end if
    end for
end for
$M \leftarrow M \setminus R$
$\mathrm{PPKI}_{OUT} \leftarrow \mathrm{PPKI}_{IN}$
for all $\mathrm{pk}_{IN} \in \mathrm{PPKI}_{IN}$ do
    if $(\mathrm{pk}_{IN}, \mathrm{pk}_{OUT}, \sigma_j) \in M$ then
        $\mathrm{PPKI}_{OUT}[\mathrm{pk}_{IN}] \leftarrow \mathrm{pk}_{OUT}$
    end if
end for
return $\mathrm{PPKI}_{OUT}$
```

To summarize:

1) Each process generates a key pair to output
2) Each process signs their output public key with their input secret key
3) Each process invokes the reliable broadcast protocol, sending their signed output key
4) Each process removes messages from keys not in the input PKI
5) Each process removes messages with signatures that do not verify.
6) Each process removes messages from public keys that sent multiple messages.
7) Each process replaces the old keys with the new keys in the PKI, and outputs the new PKI.

C. PKI Anonymization

1) Anonymous Multi-Value All-To-All Broadcast Consensus: We now specify our Anonymous Multi-Value All-To-All Broadcast Consensus protocol. In the following

- Let $v_{i,j}$ be the jth input value of process p_i.
- Let $(\mathrm{pk}_{RING,i}, \mathrm{sk}_{RING,j})$ be the key pair of process i with respect to $\mathrm{PPKI}_{RING\langle I \rangle}$.
- Let $\mathrm{pks}_{RING} = \{\mathrm{pk}_{RING,1}, \mathrm{pk}_{RING,2}, \dots, \mathrm{pk}_{RING,n}\}$ be the set of public keys in $\mathrm{PPKI}_{RING\langle I \rangle}$.
- Let rn be a unique identifier for one protocol instance (e.g. "ACv1_round15").
- Let H be a cryptographic hash function.

Algorithm 2 Anonymous Multi-Value All-To-All Broadcast Consensus Algorithm

```
for all $0 \le j \le m$ do
    $issue_j \leftarrow H(rn||j||\,\mathrm{pks}_{\mathrm{RING}})$
    $tag_j \leftarrow (issue_j, \mathrm{pks}_{\mathrm{RING}})$
    $\mathrm{rs}_{i,j} \leftarrow L.\mathsf{Sig}_{\mathrm{sk}_{RING,i}}(tag_j, \mathrm{pk}_{OUT,i})$
end for
$\mathrm{CovertBroadcastSend}_i(\mathrm{sspk}_{i,j} := (issue_j, v_{i,j}, rs_{i,j})$
while CovertBroadcastReceive do
    $\mathrm{sspk} \leftarrow \mathrm{CovertBroadcastReceive}$
    $j \leftarrow sspk.j$
    $\mathrm{sspks}_{i,j} \leftarrow \mathrm{sspks}_{i,j} \cup \{\mathrm{sspk}\}$
end while
$\mathrm{BroadcastSend}_i(\{\mathrm{sspks}_{i,j}\}_{j=0}^{m})$
for all $1 \le i' \le n$ do
    $\mathrm{sspks}_i \leftarrow \mathrm{BroadcastReceive}_{i'}$
    $\mathrm{sspkss}_j \leftarrow \mathrm{sspkss}_j \cup \{\mathrm{sspks}_{i,j}\}$
end for
for all $0 \le j \le m$ do
    $\mathrm{sspksu}_j \leftarrow \bigcup_{i=1}^{n} \mathrm{sspkss}_{i,j}$
end for
for all $0 \le j \le m$ do
    for all $\mathrm{sspk} = (\mathrm{pk}_{OUT,i}, rs_{i,j}) \in \mathrm{sspksu}_j$ do
        if $L.\mathsf{Ver}(tag_j, \mathrm{pk}_{OUT,j}, \sigma_j) = 0$ then
            $\mathrm{sspksu}_j \leftarrow \mathrm{sspksu}_j \setminus \{\mathrm{sspk}_i\}$
        end if
    end for
end for
for all $0 \le j \le m$ do
    for all $1 \le a \le n'_j$ do
        for all $1 \le b < a$ do
            if $pk_{IN,a} = pk_{IN,b}$ then
                $R \leftarrow R \cup \{m_a, m_b\}$
            end if
            if $L.Link(rs_a, rs_b)$ then
                $R \leftarrow R \cup \{m_a, m_b\}$
            end if
        end for
    end for
    $\mathrm{sspksu}_j \leftarrow \mathrm{sspksu}_j \setminus R$
end for
return $\{\mathrm{sspksu}_1, \mathrm{sspksu}_2, \dots, \mathrm{sspksu}_m\}$
```

To summarize:

1) Processes sign their values with respect to the ring and the current issue.

2) Processes anonymously broadcast their value.
3) Each process reliably broadcasts their value set.
4) Processes compute the union of the value sets.
5) Processes remove messages from the union that fail signature verification.
6) Processes identify messages which are linked to other messages in the union and remove them.
7) Processes output the remaining values.

2) Shadow PKI Protocol: We now combine the two previously defined protocols to create a complete Shadow PKI Protocol. given a linkable ring signature scheme L.

Algorithm 3 Shadow PKI Algorithm

```
$\mathrm{PPKI}_L \leftarrow$ **Algorithm1**$(L)$
$\mathrm{pk}_1, \ldots, \mathrm{pk}_m \leftarrow \mathrm{KeyGen}$
$\mathrm{sspks}_1, \ldots, \mathrm{sspks}_m \leftarrow$ **Algorithm2**$_{\mathrm{PPKI}_L}(\mathrm{pk}_1, \ldots, \mathrm{pk}_m)$
**for all** $1 \leq j \leq m$ **do**
    $\mathrm{PPKI}_{OUT,i} \leftarrow \mathrm{PPKI}_{IN}$
    $\mathrm{PPKI}_{OUT,i}[\mathrm{pk}] \leftarrow \mathrm{sspks}_i$
**end for**
**return** $\mathrm{PPKI}_{OUT}$
```

To summarize:

1) The PKI Reconstitution Protocol is used to initialize keys of all other honest processes in the ring signature PKI.
2) Processes generate keys to anonymously publish in the broadcast.
3) Processes apply the Anonymous Multi-Value All-To-All Broadcast Consensus protocol to distribute their generated keys.

Remark 8. *Critical to this protocol is the linkable ring signature primitive [20]. This primitive allows for a ring signature [13] where messages have metadata that prevents members from signing messages more than once. A more powerful primitive that may be substituted is the recently described traceable ring signature [19] the benefit of the traceable ring signature is that it allows for the identification of misbehaving parties, so that in future applications of the protocol, those parties may be ignored.*

APPENDIX C
PROOFS OF PROPERTIES OF PKI RECONSTITUTION SCHEME

A. PKI Reconstitution Scheme Liveness

Theorem 1. *The PKI Reconstitution Protocol is live*

Proof. We follow the execution of a non-faulty process through the steps of the protocol:

1) The process generates an output key pair
2) The process signs its key pairs
3) The process participates in reliable broadcasts for all key pair. Since we assume that this broadcast protocol is live, these broadcasts terminate.
4) The process removes messages from public keys not in the input PKI.
5) The process removes messages that fail signature verification.
6) The process identifies messages which are signed by the same key as other messages to other messages and removes them.
7) The process creates a new PKI by substituting the new keys for the old.
8) The process outputs the PKI

Thus, the protocol finishes with an output, so it is live. □

B. PKI Reconstitution Scheme Safety

Theorem 2. *The PKI Reconstitution Protocol is safe*

Proof. We go through the steps of the non-faulty processes through the protocol, reasoning about the execution state of each process at each step.

1) The process generates an output key pair
2) The process signs its key pairs.
3) The process broadcasts its signed key. Since we assume that this broadcast protocol is safe, each non-faulty node receives the same set of signed public keys.
4) The process removes messages from public keys not in the input PKI. Since each process starts with the same PKI all non-faulty processes remove the same values, and have a consistent set at the end of this step.
5) The process removes messages that fail signature verification. Since the verification is deterministic, all non-faulty processes remove the same values, and have a consistent set at the end of this step.
6) The process identifies messages which are signed by the same key as other messages to other messages and removes them. Since the these duplicate checks are deterministic, all non-faulty processes remove the same values, and have a consistent set at the end of this step.
7) The process creates a new PKI by substituting the new keys for the old. Since we are left with the same values on each non-faulty node, this results in the same PKI
8) The process outputs the PKI.

Thus, the protocol is live.

□

C. PKI Reconstitution Scheme Validity

Theorem 3. *The PKI Reconstitution Protocol is valid.*

Proof. Suppose process i is non-faulty. We show that process i's generated key is output in the returned PKI.

We go through the steps of the protocol.

1) Process i generates an output key pair
2) Process i signs its key pairs
3) Process i broadcasts its signed key. Thus, all non-faulty nodes now contain i's key.
4) A non faulty process removes messages from public keys not in the input PKI. Since i is in the input PKI, its key is not removed.
5) A non faulty process removes messages that fail signature verification. Since i is non faulty, it has made a valid signature, and its key is not removed.

6) A non faulty process identifies messages which are signed by the same key as other messages to other messages and removes them. Since i does not sign more than one message, its key is not removed.
7) A non faulty process creates a new PKI by substituting the new keys for the old. Since i's new key has survived, it is included.
8) A non faulty process outputs the PKI, including i's new key. Thus, the protocol is valid.

□

D. PKI Reconstitution Scheme Integrity

Theorem 4. *The PKI Reconstitution Protocol has the integrity property.*

Proof. To see this, we consider the final two steps of the PKI Reconstitution Protocol. In the penultimate step, we replace keys on the *existing identities* in the input PKI. Since no new identities are added here, the protocol is integrous. □

E. PKI Reconstitution Scheme Inclusion

Theorem 5. *The PKI Reconstitution Protocol has the inclusion property.*

Proof. To see this, again consider the final two steps of the PKI Reconstitution Protocol. In the penultimate step, we replace keys on the identities in the input PKI. Since no identities are removed here, the protocol is inclusive. □

F. PKI Reconstitution Scheme Decentralization

Theorem 6. *The PKI Reconstitution Protocol has the decentralization property.*

Proof. Note that the PKI Reconstitution Protocol is described as a single algorithm which all processes run simultaneously. While some nodes take a leader role in the broadcast subprotocol to broadcast data, ultimately, the input data for each node consists only of the PKI and the individual's private key - there is no input data distinguishing a particular node as a leader. □

G. PKI Reconstitution Scheme Synchrony

Theorem 7. *The PKI Reconstitution Protocol has the synchrony property.*

Proof. We follow the execution of a non-faulty process through the steps of the protocol, showing from the synchrony of the subprotocol that each step of the algorithm can be predetermined to finish at some fixed time.

1) The process generates an output key pair. This step requires a fixed amount of computation and no network use, so we can specify that this step takes fixed time Δ_1, and finishes at time $t_1 = t + \Delta_1$.
2) The process signs its key pairs. This step requires a fixed amount of computation and no network use, so we can specify that this step takes fixed time Δ_2, and finishes at time $t_2 = t_1 + \Delta_2$.
3) The process broadcasts its signed key. Since the reliable broadcast protocol is synchronous and initiated at the fixed time t_2, again, the node will finish this step in a fixed amount of time Δ_3 and move on. We also assume a finite cap on the number and size of messages adversaries can send, so the processing of these messages will terminate in time Δ'_3. Thus, this step takes fixed time $\Delta_3 + \Delta'_3$, and finishes at time $t_3 = t_2 + \Delta_3 + \Delta'_3$.
4) The process removes messages from public keys not in the input PKI. This step requires a fixed amount of computation and no network use, so we can specify that this step takes fixed time Δ_4, and finishes at time $t_4 = t_3 + \Delta_4$.
5) The process removes messages that fail signature verification. This step requires a fixed amount of computation and no network use, so we can specify that this step takes fixed time Δ_5, and finishes at time $t_5 = t_4 + \Delta_5$.
6) The process identifies messages which are signed by the same key as other messages to other messages and removes them. This step requires a fixed amount of computation and no network use, so we can specify that this step takes fixed time Δ_6, and finishes at time $t_6 = t_5 + \Delta_6$.
7) The process creates a new PKI by substituting the new keys for the old. This step requires a fixed amount of computation and no network use, so we can specify that this step takes fixed time Δ_7, and finishes at time $t_7 = t_6 + \Delta_7$.
8) The process outputs the PKI. Since the node has not crashed, the protocol outputs. We can specify that this step takes fixed time Δ_8, and finishes at time $t' = t_8 = t_7 + \Delta_8$.

Thus, from this analysis, there is a fixed time t' at which the protocol outputs for any non-faulty process.

□

H. PKI Reconstitution Scheme Sybil Resistance

Theorem 8. *The PKI Reconstitution Protocol has the Sybil-resistance property.*

Proof. This follows from the previous proofs of integrity and inclusivity. Again consider the final two steps of the PKI Reconstitution Protocol. In the penultimate step, we replace keys on the identities in the input PKI. Since no identities are changed, there is no possibility of a Sybil attack □

Appendix D
Proofs of Properties of the Anonymous Multi-Value All-To-All Broadcast Consensus Scheme

A. Cryptographic Properties

In order for the Anonymous Broadcast scheme we have described to be secure, the ring signature scheme which it uses must satisfy cryptographic properties. In principle, it is possible to implement our scheme using either a linkable ring signature or a traceable ring signature. The latter has the additional benefit that it can improve the resilience of the Anonymization protocol by allowing honest parties to trace the identities of malicious processes that submit more than one signature, and potentially remove them from later instantiations of the protocol. For reference, we include here definitions of properties, based on those in [19], which are sufficient to guarantee the formal security of our anonymization protocol.

Definition 16. *We define a linkability game as follows: An adversary $\mathcal{A}$ must supply a ring of n public keys, and a specific tag, along with $n+1$ signatures for that tag, all of which pass verification, and no two of which return unlinked when passed to the tracing function. A Traceable Ring Signature scheme is linkable if no PPT avdersary wins this game with non-negligible probability.*

This definition is equivalent to that given in Fujisaki and Suzuki

Definition 17. *We define an exculpability game as follows: An adversary $\mathcal{A}$ is given access to a target public key pk, as well as a signing oracle that produces signatures from pk on any message with any tag. The adversary must supply two signatures that verify correctly and trace to pk, and for which one does not trace to pk with any of the signature provided by the oracle. A Traceable Ring Signature scheme is exculpable if no PPT avdersary wins this game with non-negligible probability.*

This definition is equivalent to that given in Fujisaki and Suzuki.

Definition 18. *We define an anonymity game as follows: An adversary $\mathcal{A}$ is given access to target public keys pk_0 and pk_1, as well as a pair of signatures, one from each key, in random order, for a ring of the adversaries choosing. The adversary must identify which is which. A Traceable Ring Signature scheme is anonymous if no PPT adversary wins this game with probability non-negligibly greater than 1/2.*

This definition is not quite equivalent to that given in Fujisaki and Suzuki. In this definition, the adversary has access to two signatures, one from each unknown party, whereas in the definition of Fujisaki and Suzuki, the adversary has access to only one signing oracle from a random one of the adversaries. To see how this definition reduces to the other, consider the following reduction:

Proof. Suppose an adversary $\mathcal{A}$ in our definition is given two targets and two signed messages, and can guess which signature comes from which target with prob $1/2+\epsilon$.

Now consider an adversary $\mathcal{A}'$ to the Fujisaki Suzuki game which works as follows:

Simulate $\mathcal{A}$. For one signature, generate it from the Sig_{sk_b} oracle for the target signature, and for one key, use a random public key pk_r from the two targets For the other Signature/Key pair, flip a coin. If heads, generate an adversary key and sign a message with it, and let those be the signature/key. If tails, generate two adversary keys. Sign a message with one and use that as the signature, and use the other as the public key.

If $\mathcal{A}$ links the Sig_{sk_b} oracle signature to the pk_r, then output $b = r$. Otherwise output $b = \ r$.

Crucial here is considering what $\mathcal{A}$ does in the scenario where it is passed one Associated pk/sig and another disassociated pk/sig. Let p be the probability that $\mathcal{A}$ returns that the associated pk/sig pair are associated.

- If $r = b$ and the simulator flips heads, then $\mathcal{A}$ receives two associated pk/sig pairs. $\mathcal{A}'$ has probability $1/2+\epsilon$ of success.
- If $r = b$ and the simulator flips tails, then $\mathcal{A}$ receives one associated pk/sig pair, one disassociated pk/sig pair. $\mathcal{A}'$ has probability p of success.
- If $r = \ b$ and the simulator flips heads, then $\mathcal{A}$ receives one associated pk/sig pair, one disassociated pk/sig pair. $\mathcal{A}'$ has probability $1-p$ of success. (Because if it associates the adversaries key to the adversaries signature, it associated the Sig_{sk_b} oracle signature to the pk_r, which is wrong.)
- If $r = \ b$ and the simulator flips tails, then $\mathcal{A}$ receives two associated pk/sig pairs. $\mathcal{A}'$ has probability 1/2 of success. (Because the distributions are identical)

Since these four outcomes are equally likely, the advantage of $\mathcal{A}'$ is the average of all these success probabilities, which is $1/2+\epsilon/4$. This proves that the advantage of $\mathcal{A}'$ is non-negligibly greater than $1/2$. □

B. Anonymous Multi-Value All-To-All Broadcast Consensus Liveness

Theorem 9. *The Anonymous Multi-Value All-To-All Broadcast Consensus Protocol is live*

Proof. We follow the execution of a non-faulty process through the steps of the protocol:

1) The process signs its value.
2) The process anonymously broadcasts its value. Note that since the anonymous broadcast protocol is synchronous, the node will finish this step in a finite amount of time and move on. We also assume a finite cap on the number and size of messages adversaries can send, so the processing of these messages will terminate.
3) The process broadcasts its value set. Since the reliable broadcast protocol is synchronous, again, we assume that we assume this process happens synchronously, so the node will finish this step in a finite amount of time, and the processing of the adversarial messages is finite and will terminate.
4) The process computes the union. Since each set in the union has an upper bound on size, this takes finite time.
5) The process removes messages that fail signature verification. Since we only do one verification for each member of the set, this takes finite time.
6) The process identifies messages which are linked to other messages and removes them. Since we only do

comparisons for at most each pair in the union, this takes finite time.
7) Since the node has not crashed, the protocol outputs.

□

C. Anonymous Multi-Value All-To-All Broadcast Consensus Protocol Safety

Theorem 10. *The Anonymous Multi-Value All-To-All Broadcast Consensus Protocol is safe*

Proof. We go through the steps of the non-faulty processes through the protocol, reasoning about the execution state of each process at each step.

1) The non-faulty processes sign their value.
2) The non-faulty processes anonymously broadcast their value. Note that faulty processes may send different keys to different non-faulty nodes, or multiple keys or no keys. However, at the end of this step, each honest node i will have *some* set of keys sspks_i.
3) Each process reliably broadcasts a value set. Since we assume that the reliable broadcast primitive we are using is safe, the non-faulty processes agree on a particular value set for each process. We can term sspks_i to be the agreed upon value for i, even if i is faulty, (in which case sspks_i may be empty or nonempty). Thus, sspks_i is the same for all non-faulty processes.
4) The non-faulty processes compute the union of the sspks_i. Since the inputs to the union are the same for each, all compute the same sspksu.
5) The non-faulty processes remove messages that fail signature verification. Since the signature verification is deterministic, all non-faulty processes remove the same messages in this step, and are left with the same result.
6) The non-faulty processes identify messages which are linked to other messages and removes them. Since the signature linking is deterministic, all non-faulty processes remove the same messages in this step, and are left with the same result.
7) The non-faulty processes output. Since each of them maintained the same set through the removal process, they ultimately output the same value.

□

D. Anonymous Multi-Value All-To-All Broadcast Consensus Protocol Validity

Theorem 11. *The Anonymous Multi-Value All-To-All Broadcast Consensus Protocol is valid.*

To show this, we first prove a lemma about the inability of the adversary to attack the signature of any of the non-faulty processes

Lemma 1. *Let $\mathcal{A}$ be any PPT adversary. The probability that the adversary generates a signature such that $\mathsf{Link}(L, (\mathrm{pk}_i, \sigma_i), (\mathrm{pk}_j, \sigma_j)) = \texttt{linked}$ from information gained in the execution of the protocol, for any honest process i, is negligible.*

Proof. This follows by reduction to the exculpability property of the linkable ring signature scheme. In particular, suppose there were an adversary $\mathcal{A}$ that created a link with non-negligible probability. Note that $\mathcal{A}$ receives, in the course of the protocol, signatures from non-faulty processes for only a single value for each tag, since non-faulty processes only make a single signature in each iteration of the loop in step 1, and that signature has a different issue and tag in all iterations of the loop (assuming there are no hash collisions on H). But then we could construct an adversary $\mathcal{A}'$ which violates the exculpability property of the ring signature: $\mathcal{A}'$ simulates the Shadow PKI Protocol, with $\mathcal{A}$ acting as the faulty nodes. To simulate signatures by non-faulty nodes, $\mathcal{A}$ queries an oracle provided for signatures from honest ring signature users. Since $\mathcal{A}$ finds a link with non-negligible probability, $\mathcal{A}'$ can return these linked signatures if it finds them, and it will also succeed with non-negligible probability. □

We now prove the theorem

Proof. Suppose process i is non-faulty. We show that process i outputs its own generated public key with overwhelming probability, and therefore, by safety, all honest nodes will do the same.

We go through the steps of the protocol.

1) The non-faulty processes sign their values.
2) Process i anonymously broadcasts its signed key sspk_i to all nodes. By the validity of the anonymous broadcast protocol, all non-faulty processes (including i itself) will correctly receive this message and add sspk_i to its set.
3) Process i broadcasts sspks_i. By the validity of reliable broadcast primitive, the honest nodes agree on the sspks_i transmitted by i, containing sspk_i.
4) Each non-faulty process computes the union sspksu. Since $\mathrm{sspk}_i \in sspks_i$ and $\mathrm{sspks}_i \subseteq sspksu$, we have sspk_i in the union.
5) The non-faulty processes remove messages that do not have verifiable signatures. Since sspk_i was signed by a non-faulty process, by the verification correctness of the linkable ring signature scheme, sspk_i will not be removed in this step.
6) The non-faulty processes remove linked messages. By our lemma, the probability of the Trace function returning other than "indep" for any pair of signed messages including sspk_i is negligible. Thus, the probability sspk_i is removed in this step is negligible.
7) The value is returned.

Since sspk_i is in the union in step 5, and the probability of it being removed afterward is negligible, it will be output with overwhelming probability. □

E. Anonymous Multi-Value All-To-All Broadcast Consensus Protocol Integrity

Theorem 12. *The Anonymous Multi-Value All-To-All Broadcast Consensus Protocol has the integrity property.*

Proof. To see this, we consider the final step of the Shadow PKI Protocol, which returns $PPKI(A)_{OUT\langle\Sigma\rangle}$ consisting of only identities from the input $PPKI$. □

F. Anonymous Multi-Value All-To-All Broadcast Consensus Protocol Inclusion

Theorem 13. *The Anonymous Multi-Value All-To-All Broadcast Consensus Protocol has the inclusion property.*

Proof. To see this, we consider the final step of the Shadow PKI Protocol, which returns $PPKI(A)_{OUT\langle\Sigma\rangle}$ consisting of all identities from the input $PPKI$. Thus, every identity controlled by a non-faulty process is present in the output. □

G. Anonymous Multi-Value All-To-All Broadcast Consensus Protocol Decentralization

Theorem 14. *The Anonymous Multi-Value All-To-All Broadcast Consensus Protocol has the decentralization property.*

Proof. Note that the Anonymous Multi-Value All-To-All Broadcast Consensus Protocol is described as a single algorithm which all processes run simultaneously. While some nodes take a leader role in subprotocols to broadcast data, ultimately, the input data for each node consists only of the PPKI and the individual's private key - there is no input data distinguishing a particular node as a leader. □

H. Anonymous Multi-Value All-To-All Broadcast Consensus Protocol Synchrony

Theorem 15. *The Anonymous Multi-Value All-To-All Broadcast Consensus Protocol has the synchrony property.*

Proof. We follow the execution of a non-faulty process through the steps of the protocol, showing from the synchrony of the subprotocols that each step of the algorithm can be predetermined to finish at some fixed time.

1) Processes start at time t_1
2) The process signs its value. This step requires a fixed amount of computation and no network use, so we can specify that this step takes fixed time Δ_3, and finishes at time $t_3 = t_2 + \Delta_3$.
3) The process anonymously broadcasts its value. Since the anonymous broadcast protocol is synchronous, and initiated at the fixed time t_3 at which the previous step finishes, the node will finish this step in a fixed amount of time Δ_4 and move on. We also assume a finite cap on the number and size of messages adversaries can send, so the processing of these messages will terminate in time Δ'_4. Thus, this step takes fixed time $\Delta_4 + \Delta'_4$, and finishes at time $t_4 = t_3 + \Delta_4 + \Delta'_4$.
4) The process broadcasts its value set. Since the reliable broadcast protocol is synchronous and initiated at the fixed time t_4, again, the node will finish this step in a fixed amount of time Δ_5 and move on. We also assume a finite cap on the number and size of messages adversaries can send, so the processing of these messages will terminate in time Δ'_5. Thus, this step takes fixed time $\Delta_5 + \Delta'_5$, and finishes at time $t_5 = t_4 + \Delta_5 + \Delta'_5$.
5) The process computes the union. Since each set in the union has an upper bound on size, this takes finite time, so we can specify that this step takes fixed time Δ_6, and finishes at time $t_6 = t_5 + \Delta_6$.
6) The process removes messages that fail signature verification. Since we only do one verification for each member of the set, this takes finite time, so we can specify that this step takes fixed time Δ_7, and finishes at time $t_7 = t_6 + \Delta_7$.
7) The process identifies messages which are linked to other messages and removes them. Since we only do comparisons for at most each pair in the union, this takes finite time, so we can specify that this step takes fixed time Δ_8, and finishes at time $t_8 = t_7 + \Delta_8$.
8) Since the node has not crashed, the protocol outputs. We can specify that this step takes fixed time Δ_9, and finishes at time $t' = t_9 = t_8 + \Delta_9$.

Thus, from this analysis, there is a fixed time t' at which the protocol outputs for any non-faulty process. □

I. Anonymous Multi-Value All-To-All Broadcast Consensus Protocol Sybil Resistance

Theorem 16. *The Anonymous Multi-Value All-To-All Broadcast Consensus Protocol has the Sybil-resistance property.*

To show this, we first prove a lemma about the inability of the adversary to attack the linkability of the ring signature scheme.

Lemma 2. *Let the actions of f faulty processes be controlled by a PPT algorithm $\mathcal{A}$. Define the advantage of $\mathcal{A}$ to be the probability that $\mathcal{A}$ outputs a set $\{(\mathrm{pk}_i, \sigma_i)\}_{1\le i\le f+1}$ of $f+1$ public keys and signatures not generated by an honest node, for which any pair do not link*

$$\begin{aligned}&\mathbf{Adv}^{\textit{Sybil}}[\mathcal{A}]\\ &\quad = \Pr[\forall i, j, \mathsf{Link}(L, (\mathrm{pk}_i, \sigma_i), (\mathrm{pk}_j, \sigma_j)) = \mathtt{unlinked}]\end{aligned}$$

Then $\mathbf{Adv}^{\textit{Sybil}}[\mathcal{A}]$ is negligible in the security parameter.

Proof. This follows from the linkability property of the ring signature scheme. To see this, suppose the advantage was non-negligible for adversary $\mathcal{A}$. Consider an adversary $\mathcal{A}'$ against the linkability of the ring signature scheme which behaves as follows: It simulates the protocol using $\mathcal{A}$ for the faulty nodes. To simulate signatures by non-faulty nodes, $\mathcal{A}$ queries an oracle provided for signatures from honest ring signature users. If $\mathcal{A}$ finds a set of unlinked signatures not from non-faulty public keys, $\mathcal{A}'$ can return these linked signatures, and so it will also succeed with non-negligible probability. □

We now prove the Sybil-resistance property

Proof. We go through the steps of the protocol.

1) The non-faulty processes sign their values.
2) Non-faulty processes anonymously broadcast signed keys sspk_i to all nodes. Non-faulty nodes now have between them the non-faulty signed keys, and some set of signed keys from faulty nodes

3) Non-faulty process i broadcasts sspks_i. Non faulty nodes now have between them the non-faulty signed keys, some set of signed keys from faulty nodes (potentially including those not sent in the previous step, but broadcast now).
4) Each non-faulty process computes the union sspksu. Since $\mathrm{sspk}_i \in sspks_i$ and $\mathrm{sspks}_i \subseteq sspksu$, we have sspk_i in the union. Call $F = sspksu \setminus \{\mathrm{sspk}_i\}_{i \in \text{non-faulty}}$ the set of signed keys in this union originating from faulty nodes.
5) The non-faulty processes remove messages that do not have verifiable signatures. Call F' the set of values in F remaining after this removal.
6) The non-faulty processes remove linked messages. Call F'' the set of values in F' remaining after this removal. We claim the probability of $|F''| > f$ is negligible. To see this, note that if it were not, the faulty nodes would have generated a set of $f+1$ values, and no pair these return `unlinked` from the Link function, or else they would have been removed in the previous step.
7) Since there are $n-f$ signatures from non-faulty nodes, and at most f signatures from faulty nodes, the returned PKI has no more public keys than identities.

□

J. Anonymous Multi-Value All-To-All Broadcast Consensus Protocol Anonymity

Theorem 17. *The Anonymous Multi-Value All-To-All Broadcast Consensus Protocol is anonymous.*

We base the anonymity property of the Shadow PKI Protocol on the anonymity property of the linkable ring signature scheme.

Lemma 3. *Let the actions of f faulty processes be controlled by a PPT algorithm $\mathcal{A}$, which is given access to an unordered set of signed spk values $\{spk_i, \sigma_i\}$. Define the advantage of $\mathcal{A}$ to be the probability that $\mathcal{A}$ outputs set i, spk_i for some non-faulty i, minus $\frac{1}{n-f}$*

$$\mathbf{Adv}^{\mathit{Sybil}}[\mathcal{A}] = \Pr[i \in \texttt{non-faulty} \wedge spk = spk_i] - \frac{1}{n-f}$$

Then $\mathbf{Adv}^{\mathit{Sybil}}[\mathcal{A}]$ is negligible.

Proof. This follows from the anonymity property of the ring signature scheme. To see this, suppose the advantage was non-negligible for adversary $\mathcal{A}$. Consider an adversary $\mathcal{A}'$ against the anonymity of the ring signature scheme which behaves as follows: It generates $n-f$ shadow public keys $\{spk_i\}_{1 \le i \le n-f}$, and obtains $n-f$ signatures $\{\mathrm{sspk}_i\}_{1 \le i \le n-f}$ of arbitrary values by querying a signing oracle. It then responds to a query to identify the signature associated with a particular i value by running $\mathcal{A}$ and returning the spk_i value if is matches, otherwise it chooses some spk_j at random. From the performance of $\mathcal{A}$, $\mathcal{A}'$ has a non-negligibly-greater-than-chance probability of answering correctly, which would contradict the anonymity of the signature scheme □

We now prove the anonymity property

Proof. Suppose we could violate anonymity by constructing PPT $\mathcal{A}$ with oracle access to the data sent in this protocol execution. We can construct PPT $\mathcal{A}'$ to win the game described in the lemma: Let $\mathcal{A}'$ receive a set of $n-f$ signed keys. It now simulates the oracle access to the faulty nodes view of the protocol. To see this is possible, we go through the steps of the protocol and demonstrate that all values will be that $\mathcal{A}$ receives can be computed from the unordered list $\{\mathrm{sspk}_i\}_{1 \le i \le n-f}$ of non-faulty signed messages sspk_i:

1) The non-faulty processes sign their values. No communication happens in this step, so there is no value for the oracle to show.
2) Non-faulty processes anonymously broadcast their signed keys sspk_i to all nodes. $\mathcal{A}'$ can simulate the faulty process view in this round by concatenating the unordered set of non-faulty signed shadow public keys $\{\mathrm{sspk}_i\}_{1 \le i \le n-f}$ to whatever values the faulty nodes broadcast.
3) Non-faulty process i broadcasts sspks_i. $\mathcal{A}'$ can simulate the faulty process in the ith protocol in this round by using whatever faulty values were sent to each non-faulty process i in the previous concatenated with $\{\mathrm{sspk}_i\}_{1 \le i \le n-f}$ as leader i's value.
4) Each non-faulty process computes the union sspksu. No communication happens in this step, so there is no value for the oracle to show.
5) The non-faulty processes remove messages that do not have verifiable signatures. No communication happens in this step, so there is no value for the oracle to show.
6) The non-faulty processes remove linked messages. No communication happens in this step, so there is no value for the oracle to show.
7) Processes return a PKI. No communication happens in this step, so there is no value for the oracle to show.

□

Appendix E
Proofs of Properties of the Shadow PKI Protocol

The proofs of the properties of the PKI Anonymization now follow in a routine way from the corresponding proofs for the subprotocols that comprise it.

A. Shadow PKI Protocol Liveness

Theorem 18. *The Shadow PKI Protocol is live*

Proof. This follows from the safety of the Reconstitution Protocol and the Anonymous Multi-Value All-To-All Broadcast Consensus, as both the reconstitution and broadcast steps must terminate. □

B. Shadow PKI Protocol Safety

Theorem 19. *The Shadow PKI Protocol is safe*

Proof. We follow the steps of the protocol:

1) All processes agree on the output of the reconstitution by the safety of that protocol.
2) Processes generate keys to publish in the broadcast

3) Since all honest processes are running the same broadcast protocol with the same PKI, they agree on the same output.

□

C. *Shadow PKI Protocol Validity*

Theorem 20. *The Shadow PKI Protocol is valid.*

Proof. We follow the steps of the protocol:

1) From the validity of the reconstitution protocol, all honest processes initialize keys of all other honest processes in ring signatures PKI.
2) Processes generate keys to publish in the broadcast
3) Since any honest process with a key in the ring signature obtains a key in the shadow PKI through the broadcast protocol with high probability, and all honest nodes share the same set of outputs, the Shadow PKI Protocol is valid.

□

D. *Shadow PKI Protocol Integrity*

Theorem 21. *The Shadow PKI Protocol has the integrity property.*

Proof. This follows from the integrity of the reconstitution and broadcast: All identities in the output are present in the ring signature PKI and therefore in the input PKI. □

E. *Shadow PKI Protocol Inclusion*

Theorem 22. *The Shadow PKI Protocol has the inclusion property.*

Proof. We follow the steps of the protocol:

1) From the inclusion of the reconstitution protocol, all honest processes are included in the ring signature PKI
2) Processes generate keys to publish in the broadcast
3) Since any honest process with a key in the ring signature obtains a key in the shadow PKI through the broadcast protocol with high probability, the Shadow PKI Protocol has the inclusion property.

□

F. *Shadow PKI Protocol Decentralization*

Theorem 23. *The Shadow PKI Protocol has the decentralization property.*

Proof. Note that the Shadow PKI Protocol is described as a single algorithm which all processes run simultaneously. While some nodes take a leader role in subprotocols to broadcast data, ultimately, the input data for each node consists only of the PPKI and the individual's private key - there is no input data distinguishing a particular node as a leader. □

G. *Shadow PKI Protocol Synchrony*

Theorem 24. *The Shadow PKI Protocol has the synchrony property.*

Proof. We follow the execution of a non-faulty process through the steps of the protocol, showing from the synchrony of the subprotocols that each step of the algorithm can be predetermined to finish at some fixed time.

1) Processes start at time t_1
2) The processes run the reconstitution protocol. By the synchrony of this protocol, this step requires a fixed amount of computation and no network use, so we can specify that this step takes fixed time Δ_1, and finishes at time $t_2 = t_1 + \Delta_1$.
3) Processes generate keys to publish in the broadcast. This step requires a fixed amount of computation and no network use, so we can specify that this step takes fixed time Δ_2, and finishes at time $t_3 = t_2 + \Delta_2$.
4) The processes run the broadcast protocol. By the synchrony of this protocol, this step requires a fixed amount of computation and no network use, so we can specify that this step takes fixed time Δ_3, and finishes at time $t_4 = t_3 + \Delta_3$.
5) Since the node has not crashed, the protocol outputs. We can specify that this step takes fixed time Δ_4, and finishes at time $t' = t_5 = t_4 + \Delta_4$.

Thus, from this analysis, there is a fixed time t' at which the protocol outputs for any non-faulty process. □

H. *Shadow PKI Protocol Sybil Resistance*

Theorem 25. *The Shadow PKI Protocol has the Sybil-resistance property.*

Proof. This follows from the Sybil-resistance of the subprotocols

1) Since the reconstitution is Sybil resistant, the ring signature PKI has Sybil resistance.
2) Processes generate keys to publish in the broadcast
3) Since the broadcast protocol is Sybil-resistant and operates on a Sybil-resistant input PKI, its output is Sybil resistant.

□

I. *Shadow PKI Protocol Anonymity*

Theorem 26. *The Shadow PKI Protocol is anonymous.*

Proof. This follows from the anonymity of the broadcast step: Since it is impossible to identify nodes to keys in this step, the output PKI is anonymized. □