Sybil-Resistant Parallel Mixing

Maya Kleinstein
Hebrew University of Jerusalem

Riad S. Wahby
Carnegie Mellon University

Yossi Gilad
Hebrew University of Jerusalem

Abstract

Parallel mixing [32] is a common technique for efficiently unlinking messages from their senders' identity [6, 36–39, 44, 54]. It involves multiple servers arranged in a stratified mix-network (mixnet), each shuffling a fraction of the messages in parallel with others and then relaying them to a subsequent server. By the end of the route through the mixnet's servers, after applying each server's local shuffle, all messages are mixed together, hiding the senders' identities. Unfortunately, parallel mixing is bottlenecked by the busiest server in each mixnet stratum and does not offer a way to ensure load balancing across the servers. Thus, Sybil clients can coordinate to route their messages through one victim server in the middle of the mixnet and subsequent strata, stalling message delivery for everyone and keeping their identities hidden since their messages were already shuffled with those from other clients. This paper presents BalancedMixnet, a new protocol for load balancing clients across the servers in a parallel mix network while ensuring sender anonymity. Our protocol relies on anonymous credentials to ensure clients use a route through the mixnet that is selected uniformly at random and, at the same time, let servers verify that the message is from a valid client and prevent replay attacks. The cost of issuing and validating credentials can be easily amortized across multiple messages from the same client. We implement and evaluate BalancedMixnet, illustrating that the cost of integrating it into a parallel mixnet is modest and provides substantial benefits against Sybil attacks.

Keywords

Parallel mixing, mix network, Sybil attacks and anonymity

1 Introduction

In recent years, there has been a growing concern about the privacy of communication metadata. Namely, hiding who is sending messages to whom. One of the common techniques for hiding metadata is utilizing parallel mixing [24, 32] where clients send onion-encrypted messages through a parallel network of mixing servers (parallel mixnet). Inside the mixnet, servers are connected in a stratified topology. Each server receives a portion of the messages and processes its share of the workload concurrently with the other servers. The server decrypts messages, shuffles them, and sends them to the next server on the route. In parallel mixnets, each server appears at a different stratum on different routes to maximize each server machine's utilization. The clients choose a path through the mixnet and onion-encrypt their messages for the servers on the path. They select a server at each layer in a route that conforms with the network's stratified topology. In this manner, all mixnet servers work on parcels of the message batch but still mix all messages together. Onion encryption ensures that each server only knows a message's predecessor and successor servers, and shuffling ensures that, after several hops of mixing servers, messages are unlinked from the user who sent them, even if some of the servers were rogue.

Parallel mixing has proven very efficient; it lets the mixnet scale to handle more users by adding a proportional number of servers. It was, therefore, the basis for many recent designs for metadata private communication, e.g., [6, 36–39, 54]. These systems can indeed scale well to support a large user base, but they all assume their users choose routes honestly. Namely, that clients choose routes uniformly at random. This creates a key problem for systems that rely on parallel mixing: to enjoy the technique's scaling benefits in practice, the system must deal with Sybil attacks where clients dishonestly choose routes purposefully to degrade the mixnet's performance. If a subset of the clients were malicious, they could all choose the same server to handle their messages in the middle of the mixnet, causing that server to be particularly slow to finish processing its (disproportionate) workload. Since the mixnet is synchronous, that is, a server must wait for all messages from the servers in the previous layer to arrive before it can shuffle them together and distribute them to the next layer, this attack slows processing for the entire workload. Worse yet, anonymity makes blocking the offending clients difficult; the attacker might target a server at a later stratum in the mixnet so that its messages are already shuffled with messages from honest users, and the rogue senders are hard to trace. And, the attacker could then overload a server in subsequent strata as well to keep slowing down the whole mixnet.

By crippling performance, the attacker might drive users away from the mixnet to non-private communication alternatives. For privacy-enhancing systems, the size of the user base is crucial. Users can only hide their messages in the crowd of other users [22]. Thus, the attacker achieves two benefits: First, they compromise the privacy of users leaving the mixnet. Second, they reduce the anonymity set for users who opt to keep using the mixnet despite its poor performance while also letting them exchange fewer messages.

This work presents BalancedMixnet, a mechanism for mitigating the parallel mixnet's Sybil attack problem by issuing anonymous credentials to clients sending messages. In BalancedMixnet, clients interact with an identity provider(s) that issues them a credential. Clients then deterministically derive the route from this credential and provide efficient proof that they have a valid credential to the servers along the route. This proof hides the identity of the client holding the credential and, at the same time, ensures that clients choose honest routes and prevents them from using the same credential multiple times. The identity providers implement Sybil-resistant admission control of users to the mixnet, preventing an attacker from overloading BalancedMixnet with many fake

Proceedings on Privacy Enhancing Technologies 2025(4), 639–652

https://doi.org/10.56553/popets-2025-0149

users. The choice of admission control policy is independent of BalancedMixnet's design, and we discuss examples of privacy-friendly admission policies that keep user identities secret from the identity provider to avoid impacting user anonymity.

We faced two significant challenges in simultaneously achieving privacy and performance with BalancedMixnet. First, while clients should deterministically derive routes from their credentials, servers (including the identity provider issuing the credentials) must be unable to do so. Otherwise, those servers could follow the message's path through the mixnet and learn which client sent what message. Yet, servers must also be able to dedup messages derived from the same credential, or one rogue client using a legitimate route could just flood them with many messages. The second challenge we faced was maintaining the performance standard of non-Sybil-resistant parallel mixnets. Servers should efficiently verify that messages are on the correct route. Moreover, when there is an attack, the messages from malicious clients should be filtered *before* reaching the target server, or they would have already slowed down the network.

We address the first problem using blind signatures. This lets BalancedMixnet issue a client credential without learning the credential itself. Clients then derive the route from this credential along with a per-hop one-time ticket, which the servers then validate using a non-interactive zero-knowledge (NIZK) proof. We address the second issue, performance, by having upstream mixnet servers check that the next hop is valid rather than having each mixnet server check that they are the correct current hop; this lets BalancedMixnet filter messages with invalid routes before they reach the target server. Furthermore, we illustrate how circuit-based routing allows for mitigating the computational cost of validating tickets by amortizing it over many messages. We show that BalancedMixnet keeps users' privacy while mitigating Sybil attacks through analysis.

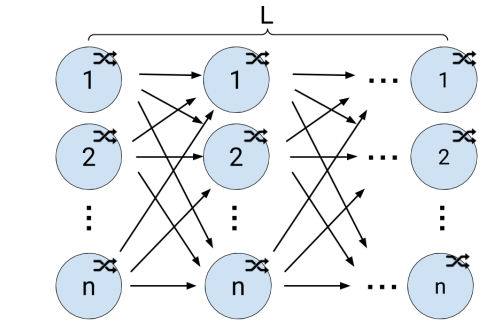
We implement BalancedMixnet in Rust and benchmark its performance. Compared to a vanilla parallel mixnet, BalancedMixnet induces an overhead only at circuit setup, which diminishes when a Sybil attack is present. After circuits are set up, we show that BalancedMixnet significantly outperforms the vanilla mixnet when there is a Sybil attack (and that performance is comparable to the vanilla when there is no attack). Sybil attacks should be expected when deploying parallel mixnets in practice, and we believe BalancedMixnet provides a tangible path towards mitigating this class of attacks and enabling real-world deployment of such systems.

2 Background

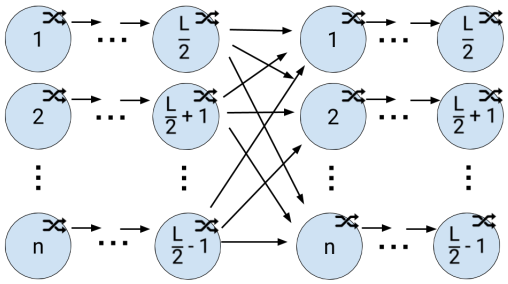
2.1 Parallel Mixing Topologies

State-of-the-art systems for metadata-private communication systems rely on different parallel mixing topologies, where the servers in a mixnet are arranged in a stratified topology.

Figure 1 illustrates two example topologies used in recent systems. The servers' topology structure allows reasoning about the network's message shuffling; e.g., certain topologies provide "perfect" shuffles, where each message submitted to the mixnet has an equal chance of being any of the outputs [36]. In contrast, other topologies use fewer strata and, therefore, achieve better performance but allow for a small bias in message output distribution,



(a) Fully-connected servers [6, 38, 39, 44, 51]



(b) Bipartite graph of cascades [54]

Figure 1: Parallel mixing topologies with n servers arranged in L layers.

which they bound to be very small [39, 54]. When a user writes a message, her client onion-encrypts it for one server in each mixnet layer. Each server decrypts the next layer in the onion, shuffles the batch of messages it receives from its predecessors, and forwards each message to the next hop on its route [18]. We architect BalancedMixnet to distribute the load of user messages independently of the underlying mixing topology; that is, BalancedMixnet ensures that at every layer, messages from each server are uniformly distributed across all of its outgoing edges to servers in the subsequent layer. This lets users obtain the privacy guarantee of the underlying mixnet while making it resilient to Sybil attacks.

2.2 Anonymous Credentials

In BalancedMixnet, an identity provider(s) admits a client into the system by issuing the user an anonymous credential. The client later uses this credential to create zero-knowledge proofs (informally, proofs that reveal nothing about the client's secrets beyond what is revealed by the statement being proved). These proofs establish that the client sends messages through a valid route. BalancedMixnet builds on the BBS+ scheme [3, 4, 14], reviewed below.

Groups and Pairings. Our scheme uses groups $\mathbb{G}_1$, $\mathbb{G}_2$ and $\mathbb{G}_T$ with prime order p and generators $g_1, l_1, l_2 \in \mathbb{G}_1, g_2 \in \mathbb{G}_2$, for which there exists a type-3 bilinear pairing [28] $\hat{e} : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$.

2.2.1 Computational Assumptions. Our work relies on the following (generally accepted) assumptions.

Generalized Decisional Diffie-Hellman [5]. The q-GDDH problem in $\mathbb{G}$ is: on input a vector $(u_1, u_2, \ldots, u_q, v_1, v_2, \ldots, v_q) \in \mathbb{G}^{2q}$ where the u_i are sampled at random, distinguish between the case where the v_i are sampled at random and the case where there exists r such that $\forall i \in \{1, \ldots, q\}, v_i = u_i^r$. The q-GDDH assumption holds in $\mathbb{G}$ if no PPT algorithm is non-negligibly better than guessing when solving the q-GDDH problem.

q-Strong Diffie-Hellman [11]. For a type-3 pairing as described above, and given a $q+3$-tuple

$$(g_1, g_1^x, \cdots, g_1^{x^q}, g_2, g_2^x) \in \mathbb{G}_1^{q+1} \times \mathbb{G}_2^2$$

the qSDH problem is to output a pair $(c, g_1^{\frac{1}{x+c}}) \in \mathbb{Z}_p \setminus \{-x\} \times \mathbb{G}_1$. The qSDH assumption holds if no PPT algorithm has a non-negligible advantage in solving the qSDH problem in $(\mathbb{G}_1, \mathbb{G}_2)$.

BBS+ Signatures. We recall the BBS+ signature scheme [3, 4, 14] for the special case of signing a single value $m \in \mathbb{Z}_p$.

Key Generation.

- Sample $(h_0, h_1) \xleftarrow{R} \mathbb{G}_1^2$, $x \xleftarrow{R} \mathbb{Z}_p$
- Set $w \leftarrow g_2^x$ and $w^* \leftarrow g_1^x$.
- Return secret key x and public key $pk = (w, h_0, h_1)$, and auxiliary data w^*. Note that w^* must be published but is not used except by the simulator algorithm [14, §4.5]; it can be validated by checking that $\hat{e}(w^*, g_2) = \hat{e}(g_1, w)$.

Blind Signing. Given public key $pk = (w, h_0, h_1)$, message $m \in \mathbb{Z}_p$ and secret key x, the following interactive protocol allows a signer holding a secret key to blindly sign a client's message $m \in \mathbb{Z}_p$:

- The client computes a Pederson commitment [43] on the message m as $C_m = h_0^{s'} h_1^m$; this is a hiding and binding commitment to m with opening s'. The client proves that the commitment is well formed via a proof that $C_m = h_0^{s'} h_1^m$, where s', m are secrets and C_m, h_0, h_1 are public.
- The signer verifies the client's proof, samples $e, s'' \xleftarrow{R} \mathbb{Z}_p$, computes $A \leftarrow (g_1 h_0^{s''} C_m)^{\frac{1}{e+x}} = (g_1 h_0^{s'+s''} h_1^m)^{\frac{1}{e+x}}$, and sends (A, e, s'') to the client.
- The signature on m is (A, e, s), where $s = s' + s''$.

Verification. For a public key $pk = (w, h_0, h_1) \in \mathbb{G}_2 \times \mathbb{G}_1^2$ and message $m \in \mathbb{Z}_p$, signature $(A, e, s) \in \mathbb{G}_1 \times \mathbb{Z}_p^2$ is valid iff

$$\hat{e}(A, w g_2^e) = \hat{e}(g_1 h_0^s h_1^m, g_2)$$

The BBS+ signature scheme is existentially unforgeable under the qSDH assumption [14]. The signer learns no information about m and the client is bound to m except with negligible probability; informally, the proof reveals only that C_m is well formed, and is sound if computing discrete logs is hard in $\mathbb{G}_1$ [47]. Looking ahead (§4.3), we will use the fact that the BBS+ signature allows the client to prove possession of a signature without revealing it.

3 Overview

This section gives an overview of our setting, threat model, and security definitions. We describe BalancedMixnet's design in Section 4 and analyze its security in Section 5. Figure 2 illustrates the actors in the system: the clients, the identity provider, and the mix servers. In BalancedMixnet, *clients* deliver messages through *layers* of *mix servers*. Before clients send messages, they register with the *identity provider*, which issues them an *anonymous credential*. (For simplicity, our descriptions assume a single identity provider; in §4.5, we revisit this and show how the identity provider can be distributed across multiple servers.)

As in a standard mixnet, clients onion-encrypt their messages such that each server can decrypt one layer and determine where to route the message next. In contrast to a standard mixnet, however, clients in BalancedMixnet cannot arbitrarily choose the path their message takes through the mixnet. Instead, each client uses its anonymous credential to deterministically derive a sequence of *tickets*, one per hop, that dictates the message's path. In each layer of the onion-encrypted message, the client includes both the ticket and a non-interactive zero-knowledge proof establishing that the client knows a valid credential and that the ticket was correctly derived from it. This lets servers discard messages if the next hop is inconsistent with the ticket or the ticket proof is invalid. This means that malicious messages never reach any server that is not actually on the message's path: if rogue clients send messages intended to overload a server in the mixnet, such messages are effectively filtered *in parallel* by the predecessors of the victim server. And since tickets are deterministically derived from the credentials, servers can use them to deduplicate messages from a given client, ensuring that a valid credential cannot be used to send many messages at once.

3.1 Sybil resistant admission control

The identity provider implements Sybil-resistant admission control, preventing an attacker from flooding the system with many rogue clients. It is important to note that such admission control *does not* inherently come at the cost of anonymity: the identity provider can safely admit users without knowing their real-world identities. For example, users may be required to make an anonymous payment (say, using a cryptocurrency) when registering with the identity provider, making a Sybil attack economically infeasible (this is roughly similar to the approach used by Nym [51]). Alternatively, the identity provider might require a privacy-preserving proof of personhood [12], e.g., a zero-knowledge proof establishing ownership of a valid passport [46, 50] that has not previously been used to register with the system. Other possible approaches include proof of work or using CAPTCHAs. The choice of admission control is a policy decision that is orthogonal to BalancedMixnet's design and will likely differ across deployments. Crucially, BalancedMixnet's use of anonymous credentials ensures that the identity provider does not learn which user is associated with an issued credential. The identity provider is trusted only to correctly enforce the admission-control policy of its mixnet deployment.

3.2 Communication model

Like many other mixnets, BalancedMixnet mitigates timing attacks by operating in synchronous *rounds*. Every round, each server collects messages it received from all servers in the previous layer, decrypts a layer of the onion, validates their tickets, shuffles the messages, and forwards them to the next hop. The last server delivers the message to the destination.

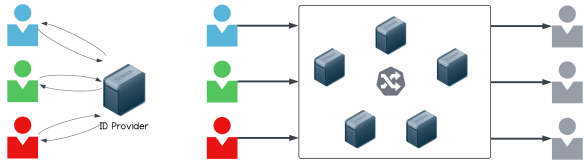


Figure 2: An overview of BalancedMixnet deployment. Users first register with the identity provider to receive a credential (left). They can then send messages through the mixnet via routes that are deterministic, yet uniformly distributed (right). The mixnet servers can check that the messages travel through a valid route, although they don't learn anything about non-adjacent hops.

Epochs and circuits. BalancedMixnet amortizes the cost of validating NIZK proofs by keeping the clients' routes fixed for an epoch, which we envision to be on the order of minutes. Similarly to Tor [23], every epoch clients send setup messages to establish efficient circuits for routing messages through the mixnet. The setup messages include the tickets and NIZK proofs discussed above; they also associate a fresh pseudonymous connection ID and symmetric key at every hop. In the following messaging rounds, clients send data messages efficiently encrypted using those symmetric keys.

3.3 Security properties

BalancedMixnet assumes a powerful adversary that can monitor the network, control some of the system's servers, and introduce many (but not infinitely many) rogue clients. BalancedMixnet has two high-level goals: maintaining the mixnet's privacy guarantee (i.e., unlinking the users' identities from their messages) and efficiently ensuring availability in the face of Sybil attacks.

To achieve these goals, BalancedMixnet leverages an *anonymous ticketing* scheme, which we define immediately below.

3.3.1 Anonymous ticketing schemes. An anonymous ticketing scheme comprises the following six randomized algorithms, all of which have access to a set of public system parameters:

- $\mathsf{IdPKeygen}() \to (\mathsf{sk}_{\mathsf{Id}}, \mathsf{pk}_{\mathsf{Id}})$ samples an identity-provider key-pair.
- $\mathsf{IdPSign}(\mathsf{sk}_{\mathsf{Id}}, \mathsf{req}_{\mathsf{Id}}) \to \mathsf{resp}_{\mathsf{Id}} \mid \perp$ processes an identity-creation request, returning a credential response or an error if the request was malformed.
- $\mathsf{CredReq}() \to (\mathsf{req}_{\mathsf{Id}}, \mathsf{aux}_{\mathsf{Id}})$ samples an identity-creation request and auxiliary information.
- $\mathsf{CredGen}(\mathsf{resp}_{\mathsf{Id}}, \mathsf{aux}_{\mathsf{Id}}) \to \mathsf{cred}$ processes an identity-creation response and the auxiliary information from the corresponding request, and outputs an anonymous credential.
- $\mathsf{TicketGen}(\mathsf{cred}, \mathsf{LayerId}, \mathsf{EpochId}) \to (t, \pi)$ creates a ticket and a proof that the ticket was correctly generated from a valid credential.
- $\mathsf{TicketVerify}(\mathsf{pk}_{\mathsf{Id}}, t, \pi) \to \mathsf{OK} \mid \perp$ checks a ticket's validity.

An anonymous ticketing scheme is *correct* if $\forall \mathsf{LayerId}, \mathsf{EpochId}$

$$\Pr\left[\begin{array}{r}(\mathsf{sk}_{\mathsf{Id}}, \mathsf{pk}_{\mathsf{Id}}) \leftarrow \mathsf{IdPKeygen}() \\ (\mathsf{req}_{\mathsf{Id}}, \mathsf{aux}_{\mathsf{Id}}) \leftarrow \mathsf{CredReq}() \\ \mathsf{resp}_{\mathsf{Id}} \leftarrow \mathsf{IdPSign}(\mathsf{sk}_{\mathsf{Id}}, \mathsf{req}_{\mathsf{Id}}) \\ \mathsf{cred} \leftarrow \mathsf{CredGen}(\mathsf{resp}_{\mathsf{Id}}, \mathsf{aux}_{\mathsf{Id}}) \\ (t, \pi) \leftarrow \mathsf{TicketGen}(\mathsf{cred}, \mathsf{LayerId}, \mathsf{EpochId}) \\ \mathsf{OK} = \mathsf{TicketVerify}(\mathsf{pk}_{\mathsf{Id}}, t, \pi)\end{array}\right] \approx 1$$

where the probability is over the algorithms' random choices.

3.3.2 Privacy. At a high level, we require BalancedMixnet to offer the same privacy guarantee as a standard mixnet: an attacker who does not control every server along a given user's path cannot discern which messages belong to that user. In BalancedMixnet, a user's messages include a ticket and proof output by TicketGen. We, therefore, require the following two properties from the anonymous ticketing scheme: *anonymity* and *unlinkability*.

Anonymity. Roughly speaking, we require that the ticket and proof output by TicketGen reveal nothing about the user's identity *except* the correct next hop. This is formalized via a standard *simulation*-based definition of zero knowledge with respect to the user's credential. Specifically, we say that an anonymous ticketing scheme satisfies anonymity if there exists a PPT algorithm Sim such that for any valid cred (i.e., one resulting from the correct sequence of invocations of CredReq, IdPSign, and CredGen), $\forall \mathsf{LayerId}, \mathsf{EpochId}$,

$$\{\mathsf{TicketGen}(\mathsf{cred},\mathsf{LayerId}, \mathsf{EpochId})\} \approx_c \{\mathsf{Sim}(\mathsf{LayerId}, \mathsf{EpochId}, \mathsf{NextHop})\}$$

That is, Sim induces a distribution (computationally) indistinguishable from TicketGen *without* access to cred, and thus a ticket and proof reveal nothing about the user's credential except NextHop.[1]

Unlinkability. Roughly speaking, we require that it is infeasible to determine if any set of ticket-proof tuples corresponds to a single user's identity or to many different identities. This is formalized via the following game between challenger C and adversary $\mathcal{A}$, in which $\mathcal{A}$ can make any (polynomial) number of queries in any order.

- **Initialize**: C samples

$$\begin{aligned} b &\xleftarrow{\$} \{0,1\} \\ (\mathsf{sk}_{\mathsf{Id}}, \mathsf{pk}_{\mathsf{Id}}) &\leftarrow \mathsf{IdPKeygen}() \\ (\mathsf{req}_{\mathsf{Id}}, \mathsf{aux}_{\mathsf{Id}}) &\leftarrow \mathsf{CredReq}() \\ \mathsf{cred} &\leftarrow \mathsf{CredGen}(\mathsf{IdPSign}(\mathsf{sk}_{\mathsf{Id}}, \mathsf{req}_{\mathsf{Id}}), \mathsf{aux}_{\mathsf{Id}}) \end{aligned}$$

  C then sends $\mathsf{pk}_{\mathsf{Id}}$ to $\mathcal{A}$.

- **Registration Query**: $\mathcal{A}$ sends a query $\mathsf{req}_{\mathsf{Id},j}$. C responds with $\mathsf{resp}_{\mathsf{Id},j} \leftarrow \mathsf{IdPSign}(\mathsf{sk}_{\mathsf{Id}}, \mathsf{req}_{\mathsf{Id},j})$.

- **Ticket Query**: $\mathcal{A}$ sends a query $(\mathsf{LayerId}_i, \mathsf{EpochId}_i)$. When $b = 0$, C computes

$$(t_i, \pi_i) \leftarrow \mathsf{TicketGen}(\mathsf{cred}, \mathsf{LayerId}_i, \mathsf{EpochId}_i)$$

[1] BalancedMixnet uses a non-interactive zero-knowledge proof in the random oracle model. The zero-knowledge property requires Sim to *program* this random oracle; this is standard in practical constructions of non-interactive zero knowledge [27, 56].

When $b = 1$, C computes

$$(\mathsf{req}_{\mathsf{Id},i}, \mathsf{aux}_{\mathsf{Id},i}) \leftarrow \mathsf{CredReq}()$$
$$\mathsf{cred}_i \leftarrow \mathsf{CredGen}(\mathsf{IdPSign}(\mathsf{sk}_{\mathsf{Id}}, \mathsf{req}_{\mathsf{Id},i}), \mathsf{aux}_{\mathsf{Id},i})$$
$$(t_i, \pi_i) \leftarrow \mathsf{TicketGen}(\mathsf{cred}_i, \mathsf{LayerId}_i, \mathsf{EpochId}_i)$$

C sends (t_i, π_i) to $\mathcal{A}$.

- **Output**: $\mathcal{A}$ outputs $b^\star \in \{0, 1\}$.

We say that an anonymous ticketing scheme satisfies unlinkability if, for all PPT adversaries $\mathcal{A}$, $\left|\Pr\left[b = b^\star\right] - 1/2\right|$ is negligible, i.e., $\mathcal{A}$ does negligibly better than guessing.

3.3.3 Availability. To ensure availability, BalancedMixnet must ensure that messages are uniformly distributed across all first-layer servers, and are then uniformly distributed across each mix server's outgoing links on all mixnet layers. (BalancedMixnet can also support essentially arbitrary nonuniform distributions; we discuss briefly in §4.2.) These properties must hold even if the attacker controls many clients and crafts messages without following the protocol. We capture these requirements via *unforgeability* and *sybil resistance*.

Unforgeability. We require that credentials and tickets in an anonymous ticketing scheme can be created only by clients permitted by the identity provider. More precisely, we say that an anonymous ticketing scheme is *unforgeable* if two properties hold: first, cred must be existentially unforgeable (i.e., EUF-CMA, the standard notion of security for a digital signature scheme). Second, there must exist a PPT algorithm Extract that outputs a valid credential cred (with high probability) when given oracle access to a client capable of generating (t, π) where $\mathsf{TicketVerify}(\mathrm{pk}_{\mathsf{Id}}, t, \pi) = \mathsf{OK}$.[2] Intuitively, this ensures that only a client that holds a valid credential can create a convincing ticket and proof.

Sybil Resistance. To ensure that users cannot flood the network with messages, we require three related properties, which together comprise *sybil resistance*. First, it must be infeasible to generate two distinct, valid tickets for a given cred, LayerId, and EpochId. Second, any message that does not correspond to a valid ticket must be rejected. And third, for every honest server in the mixnet, the outgoing messages distribute uniformly (or according to a specified distribution) across the server's links to other mix servers.[3]

Intuitively, this property guarantees that no internal node in the mixnet will handle a disproportionate number of clients. The nodes in the first layer of the mixnet are a special case because clients can always attempt to simply flood them with requests. We note, however, that servers in the first layer can use standard countermeasures (e.g., CAPTCHAs, IP reputation, etc.) when under attack since clients connect directly to these servers.

[2]As is standard in practical constructions of proofs of knowledge, the Extractor is assumed to have *rewinding* access to the prover, i.e., it can selectively restart the prover's execution at an earlier point in the computation.

[3]Note that in typical mixnets, all servers in each layer have the same number of incoming and outgoing links (as in the topologies in Figure 1). In this case, the sybil resistance property promises a uniform distribution of messages over all possible mixnet routes.

4 Design

We describe BalancedMixnet following the illustration in Figure 2. When users join the system, their clients interact with the identity provider to receive anonymous credentials (§4.1). Then, they use these credentials to derive the routes through the mixnet and set up circuits (§4.2). Clients prove to mix servers the circuit's route is correct (§4.3). Finally, the clients send messages through those circuits (§4.4). We conclude this section by discussing ways that BalancedMixnet can take advantage of multiple providers (§4.5).

System parameters. We assume that each client knows the mixnet's topology, in particular, it knows M, the number of mixes, as well as the public keys of its mix servers $pk_1, \cdots, pk_M$ so that it can choose valid routes and onion-encrypt messages. Also, we assume that everyone uses the same groups and generators for the BBS+ signature scheme (§2.2), and cryptographic hash functions $H : \{0,1\}^* \to \mathbb{Z}_p$ and $H_b : \{0,1\}^* \to \mathbb{G}_1$, which we model as random oracles.

Instantiating the anonymous ticketing scheme. BalancedMixnet's anonymous ticketing scheme builds on BBS+ signatures (§2.2.1). In particular, IdPKeygen is the BBS+ key-generation procedure. The three steps in the BBS+ blind-signing procedure correspond to CredReq, IdPSign, and CredGen, respectively. We detail the TicketGen and TicketVerify constructions below.

4.1 User Registration

After registering with the system (e.g., paying a subscription fee), the user's client chooses a random user-ID $m \xleftarrow{R} \mathbb{Z}_p$ and interacts with the identity provider to receive a blind BBS+ signature on m to be used as the anonymous credential for accessing the mixnet. To enforce expiration dates on credentials, BalancedMixnet uses key rollover. Once every several epochs (say, a fixed number matching the subscription interval, e.g., a month), the identity provider rotates its key, which invalidates old credentials.

To make sure that not all users refresh their credentials at once when the key rotates, the identity provider creates a new key halfway through the time limit on the current key. This lets users ask for new credentials ahead of time (though they will still use their old credentials until the epoch when the identity provider's matching key is retired).

4.2 Circuit Setup

At the start of every epoch, the client forms a circuit. It derives the circuit's route through the mixnet from their credential and epoch ID, which is an ever-incrementing counter (stored as a 64-bit integer that we assume will never wrap). For each hop on the route, the client creates a ticket proving the correctness of the next link traversal. The client also creates a ticket for "hop zero," i.e., the choice of which entry server it connects to, which lets that server filter messages when it receives an unbalanced portion of the user load.

Using its anonymous credential (A, e, s) and blinded user id m (§2.2, §4.1), the client calculates tickets t_k for all circuit LayerIds $k \in [L]$. First, it computes a basis for each layer:

$$b_k = H_b(k \,||\, \mathsf{EpochId})$$

The client then sets the ticket to be $t_k = b_k^{e+m}$ (an element in $\mathbb{G}_1$). Notice that this ticket is derived deterministically from the credential.

Assume that the current server has ℓ outgoing links to the next layer; then the next hop $hop_{k+1} \in [\ell]$ is selected uniformly at random by computing $hop_{k+1} = H(t_k) \bmod \ell$. We note that it is straightforward to sample from other distributions, e.g., in the case that some servers can handle more traffic than others. As a simple example, to give one server more weight, one might instead reduce mod $\ell + 1$ and assign to the weightier server when the result is ℓ.

The circuit setup message then includes for each layer k:

- The ticket and proof (t_k, π_k); we describe the details of proof generation and verification in §4.3 below.
- An ephemeral random connection ID $ConnID_k$.
- A symmetric key s_k, which the client will later use to create onion messages.

The setup message is onion-encrypted by the public keys of the servers on the route through the mixnet in reverse order, such that the server at layer k can only read $(t_k, \pi_k, ConnID_k, s_k)$. That server then verifies π_k to ensure the ticket is correct and uses t_k to derive the next server, as described above. The server deduplicates circuit setup messages using t_k; since all circuit setup messages with the same credential will derive the same ticket, this prevents multiple users from using the same credential. Lastly, the server stores a mapping $\langle ConnID_k : s_k, \textit{next-hop} \rangle$ for the rest of the epoch.

4.3 Proof Instantiation

We construct proof that shows that the sender knows a valid BBS+ blind signature credential (A, e, s) signed on message $m \in \mathbb{Z}_p$ as described in 2.2, such that their ticket t is valid with respect to a known basis $b \in \mathbb{G}_1$, i.e., $t = b^{e+m}$. We use the same parameters and notation as in Sections 2.2 and 4.

4.3.1 Proof generation. The client's proof has two parts. First, it proves that it owns a valid credential without revealing that credential. Second, it proves that it derived the ticket correctly from that credential. Our approach extends the BBS+ proof of knowledge of a signature [14, §4.5] to establish both statements in one proof.

To establish credential ownership, the client samples

$$r_1 \xleftarrow{R} \mathbb{Z}_p^* \qquad r_2 \xleftarrow{R} \mathbb{Z}_p$$

and computes

$$\begin{aligned} A' &\leftarrow A^{r_1} & \overline{A} &\leftarrow (A')^{-e} \cdot (g_1 h_0^s h_1^m)^{r_1} \\ r_3 &\leftarrow r_1^{-1} & s' &\leftarrow s - r_2 \cdot r_3 \\ d &\leftarrow (g_1 h_0^s h_1^m)^{r_1} \cdot h_0^{-r_2} \end{aligned}$$

Notice that, for a valid credential, $\overline{A} = (A')^x$, where x is the identity provider's secret key.

To establish the correctness of the ticket, the client samples

$$\rho_1 \xleftarrow{R} \mathbb{Z}_p \qquad \rho_1 \xleftarrow{R} \mathbb{Z}_p$$

and computes

$$\beta_1 \leftarrow \rho_1(e+m) \qquad \beta_2 \leftarrow \rho_2(e+m) \qquad C \leftarrow l_1^{\rho_1} l_2^{\rho_2}$$

Recall from §2.2 that l_1, l_2 are fixed public generators of $\mathbb{G}_1$.

Finally, the prover generates a zero-knowledge proof of knowledge Π with secret inputs $(m, e, r_2, r_3, s', \rho_1, \rho_2, \beta_1, \beta_2)$ and public inputs $(A', \overline{A}, d, C, t, b, g_1, h_0, h_1, l_1, l_2)$ establishing the relations

$$\begin{aligned} \overline{A} d^{-1} &= A'^{-e} \cdot h_0^{r_2} & g_1 &= d^{r_3} h_0^{-s'} h_1^{-m} \\ C &= l_1^{\rho_1} l_2^{\rho_2} & C^{e+m} &= l_1^{\beta_1} l_2^{\beta_2} \\ b^{\beta_1} &= t^{\rho_1} \end{aligned}$$

We note that the first two relations are inherited from the BBS+ signature; the others establish correctness of the ticket.

The client renders the proof Π non-interactive via the (strong) Fiat-Shamir heuristic, i.e., the challenge computation includes all system parameters:

$$h_{\text{params}} = H(M, w, pk_1, \ldots, pk_m, EpochID, k, b, g_1, g_2, h_0, h_1, l_1, l_2)$$

In more detail, the proof Π is computed as follows:

Commit. Sample $r_m, r_e, r_{r_2}, r_{r_3}, r_{s'}, r_{\rho_1}, r_{\rho_2}, r_{\beta_1}, r_{\beta_2} \xleftarrow{R} \mathbb{Z}_p^*$ and compute

$$\begin{aligned} T_1 &\leftarrow A'^{-r_e} \cdot h_0^{r_{r_2}} & T_2 &\leftarrow d^{r_{r_3}} h_0^{-r_{s'}} \\ T_3 &\leftarrow l_1^{r_{\rho_1}} l_2^{r_{\rho_2}} & T_4 &\leftarrow C^{-(r_e + r_m)} l_1^{r_{\beta_1}} l_2^{r_{\beta_2}} \\ T_5 &\leftarrow b^{r_{\beta_1}} t^{-r_{\rho_1}} \end{aligned}$$

Challenge. Invoke the random oracle on the system parameters, public inputs, and commitments:

$$c \leftarrow H(h_{\text{params}}, (A', \overline{A}, d, C, t), (T_1, T_2, T_3, T_4, T_5))$$

Response. Compute

$$\begin{aligned} s_m &\leftarrow r_m - cm & s_e &\leftarrow r_e - ce & s_{r_2} &\leftarrow r_{r_2} - cr_2 \\ s_{r_3} &\leftarrow r_{r_3} - cr_3 & s_{\rho_1} &\leftarrow r_{\rho_1} - c\rho_1 & s_{\rho_2} &\leftarrow r_{\rho_2} - c\rho_2 \\ s_{\beta_1} &\leftarrow r_{\beta_1} - c\beta_1 & s_{\beta_2} &\leftarrow r_{\beta_2} - c\beta_2 \end{aligned}$$

Output. The proof Π is:

$$\Pi = (c, s_m, s_e, s_{r_2}, s_{r_3}, s_{\rho_1}, s_{\rho_2}, s_{\beta_1}, s_{\beta_2})$$

4.3.2 Verification. To verify a proof Π that $(A', \overline{A}, d, C, t)$ comprise a valid credential and ticket for basis b, the verifier checks that $A' \neq 1_{\mathbb{G}_1}$ and $\hat{e}(A', w) = \hat{e}(\overline{A}, g_2)$, then verifies Π as follows:

Compute.

$$\begin{aligned} T_1' &\leftarrow (A')^{-s_e} \cdot h_0^{s_{r_2}} \cdot \left(\overline{A} d^{-1}\right)^c & T_2' &\leftarrow d^{s_{r_3}} h_0^{-s_{s'}} h_1^{s_m} \cdot g_1^c \\ T_3' &\leftarrow l_1^{s_{\rho_1}} l_2^{s_{\rho_2}} C^c & T_4' &\leftarrow C^{-(s_e + s_m)} l_1^{s_{\beta_1}} l_2^{s_{\beta_2}} \\ T_5' &\leftarrow b^{s_{\beta_1}} t^{-s_{\rho_1}} \end{aligned}$$

Output. Compute

$$c' = H(h_{\text{params}}, (A', \overline{A}, d, C, t), (T_1', T_2', T_3', T_4', T_5'))$$

Return OK if $c' = c$, else $\perp$.

As in [14], this construction does not work for $A = 1_{\mathbb{G}}$; this happens with negligible probability for an honest signer.

4.4 Sending Messages

After circuit setup, clients send data messages on the circuits they have established. Since at circuit setup the client registered at every mix server along the route an ephemeral connection ID, it can now use this ID to reference a secret key and the next hop server. In every message, the client includes its data, prepended with the ephemeral IDs for each hop in reverse order, and onion encrypted with its installed symmetric keys. This lets communication rounds be very efficient, since they rely on symmetric cryptography.

Every circuit can carry one message per round. When a server receives messages from the servers connected to it at the previous hop, it deduplicates them according to the connection ID. It then fetches the matching symmetric key and next hop server for that message, and decrypts its layer of the (symmetric) onion. Once it receives all messages from its predecesors in the topology, the server shuffles the messages and sends each message to its next hop.

4.5 Multiple Identity Providers

Thus far, we have described BalancedMixnet with just one identity provider. However, a single such provider would be a single point of failure. If that server is down, then users cannot receive credentials. Moreover, if that identity provider is malicious, it could selectively deny service from some users (forcing them to use another, e.g., non-private, communication method) or only issue credentials to a few users (to minimize the user base and, therefore, the admitted users' anonymity set). On the other hand, a rogue malicious identity provider can just create as many credentials as the attacker may want, without asking to authenticate or pay a fee for using the system, which allows a Sybil attack.

We address this challenge in BalancedMixnet using a recent threshold version of the BBS+ anonymous credential scheme [25]. Using threshold BBS+, we split the identity provider into n servers and require that $t + 1$ of them must consent to issuing a client's certificate. On setup, the n servers jointly create a new public key, and each of the servers obtains a share of the secret key (without ever learning the secret). Key generation is performed through a distributed protocol, such that no server sees or learns information about more than a single shard of the secret key (see [25, §9] for a full description of such a protocol). Although the threshold version of BBS+ distributes trust in the identity provider across multiple servers, there is still a single BBS+ signing key (distributed across the servers) that governs the certificate issuance. To improve security against the risk of exposure of its shards over time, the signing key may be rotated periodically, which involves running the distributed key generation protocol per rotation. The distributed key generation protocol for BBS+ is efficient, with the most recent proposals illustrating that the protocol run-time is on the order of tens of milliseconds to several seconds for 10-20 servers [26, 57] (depending on whether robustness against corrupt servers is required). We expect that changes to the server set or key rotation would happen much less frequently than the protocol's run-time interval (under either configuration), making distributed key generation practical.

In the threshold signing protocol, a client requests a credential from all n identity servers. The servers then exchange two messages between themselves and provide a response to the client. If the client receives a response from $t + 1$ servers, it can reconstruct a valid credential. The threshold BBS+ variant supports any $t < n$, and a deployment's choice of t should balance security against both types of attacks above. Namely, no minority of malicious (or faulty) identity servers can prevent a legitimate client from obtaining a credential, nor can they issue certificates without properly authenticating the users.

4.5.1 Deployment tradeoffs. BalancedMixnet's design is agnostic to the identity providers' deployment, as long as it satisfies the threshold BBS+ security assumption (at most t dishonest servers). This gives a spectrum of decentralization vs. performance tradeoff points. On one end of the spectrum, the mix servers themselves run the identity servers and jointly act as the distributed identity provider. This option allows tuning the threshold BBS+ security parameter to match the number of assumed dishonest mixnet servers, thereby avoiding the introduction of further security assumptions on the honesty of identity providers. Another benefit of this choice is that the identity provider's availability is linked to that of the mixnet. Namely, if many of the identity providers' servers are down, then many mix servers are also down, so users might not be able to use the mixnet anyways (put differently, the reliance of BalancedMixnet on an identity provider wouldn't impact the overall mixnet availability). If there are many servers in the mixnet, the above deployment choice would slow down credential generation since there would be many participant in the threshold BBS+ signing protocol, but we only run once per user; for moderate-size deployments (e.g., tens of servers), we believe this deployment choice provides a good trade-off. Large deployments could sample a subset of the mix servers to run the distributed identity provider, limiting the overhead of the threshold protocol. Consider the simple strategy where the mix servers operating the identity provider are selected independently at random out of all mixnet servers. In this case, the chance that the sampled set has a significant deviation in the rate of malicious servers from their rate in the overall mixnet quickly diminishes with the sample size and deviation size (Chernoff bound). Therefore, the sampling approach is particularly suitable for such large deployments, as it allows for tightly setting the threshold BBS+ security parameter while limiting the number of participants in the signing protocol for performance.

On the other end of the spectrum are deployments where servers in the mixnet frequently changes, like in Nym [51] or Tor [23], making the mix servers unsuitable to double as identity servers. For example, a deployment may permissively allow organizations to contribute servers to the mixnet, but such servers may have a relatively high churn rate or illustrate malicious behavior (like corrupting user messages), and be removed from the network (e.g., Tor monitors its servers' uptime and checks that they don't corrupt the content they relay [30]). If the set of servers changes with the admission or removal of each mix server, the identity providers' distributed key generation for the threshold BBS+ signature key may need to run often, which incurs communication and computation costs for the servers. Moreover, users would need to reissue their credentials for the new BBS+ key, which may burden adoption. For

such mixnet deployments, using a fixed set of dedicated identity provider servers is more suitable.

5 Analysis

In this section we show that BalancedMixnet achieves its privacy and availability goals as defined in §3.3.

5.1 Proof Protocol Properties

The proof protocol detailed in Section 4.3 is the result of applying the Fiat-Shamir heuristic to a complete, zero-knowledge interactive proof of knowledge. In particular, that interactive proof extends the BBS+ proof of knowledge of signature protocol [14, §4.5] by adding three additional relations (§4.3.1). The protocol is a generalized Schnorr proof [15]; its simulator and extractor are standard.

At a high level, the simulator for the interactive version of a generalized Schnorr proof works by sampling random proof elements, then computing the prover's initial messages in a way that ensures consistency with the proof elements (i.e., by solving the verification equations). The extractor works by running the prover to produce an accepting transcript; rewinding the prover and re-running with the same commitments and a different challenge; and then solving the system of linear equations induced by the two transcripts. The conversion from an interactive to a noninteractive proof using the Fiat-Shamir heuristic preserves simulation and extractability.

Putting it all together: the protocol of Section 4.3 is complete by inspection, and its simulator and extractor are the standard ones described immediately above.

5.2 Privacy

We first show that BalancedMixnet's anonymous ticketing scheme provides Anonymity (§3.3.2). In particular, the anonymous ticketing scheme has a PPT algorithm Sim that, on input LayerId, EpochId, and NextHop, outputs a corresponding ticket and convincing proof. Sim closely follows the one from BBS+ [14, §4.5]: it computes $b \leftarrow H_b(\mathsf{LayerId} \,||\, \mathsf{EpochId})$ and samples

$$\rho \xleftarrow{R} \mathbb{Z}_p^* \qquad\qquad (d, C) \xleftarrow{R} \mathbb{G}_1^2$$

Next, the simulator samples $t \xleftarrow{R} \mathbb{G}_1$ until $H(t) \equiv \mathsf{NextHop} \bmod \ell$, where ℓ is the number of servers in the successor to LayerId. (Since H is a random oracle, the probability of success in each trial is $1/\ell$; by a Chernoff bound, the simulator succeeds with overwhelming probability after, say, 100ℓ samples.) It then computes

$$A' \leftarrow g_1^{\rho} \qquad\qquad \overline{A} \leftarrow (w^*)^{\rho}$$

where w^* is the auxiliary data output by the identity provider's key generation algorithm. Finally, it runs the proof protocol's simulator for the public inputs $(A', \overline{A}, d, C, t, b)$, yielding a simulated proof Π. We note that Sim can also simulate a proof for a specified ticket t; we will use this fact below.

A simulated ticket and proof is computationally indistinguishable from a real one: first, as in the BBS+ simulator, the distribution of $(A', \overline{A}, d)$ is perfectly simulated. Likewise, the distribution of C is perfectly simulated (it is uniformly random in both cases). For the distribution of t, we have by construction that it corresponds to the correct next hop. Our remaining obligation is to show that a real ticket t is indistinguishable from uniformly random. To see why this holds, notice that the proof elements in a real proof are uniformly random and independent of $e + m$; for random t, the distribution of $\overline{e + m} = \mathrm{dlog}_b t$ in the simulated proof is likewise.

We now show that BalancedMixnet's anonymous ticketing scheme provides unlinkability (§3.3.2) by showing that $\mathcal{A}$ breaking unlinkability breaks the q-GDDH assumption on $\mathbb{G}_1$ (§2.2.1) in the programmable random-oracle model. The reduction $\mathcal{R}$ is constructed as follows. (Note that because $\mathcal{R}$ is constructed in the programmable ROM, it must answer queries to the random oracles H and H_b in addition to $\mathcal{A}$'s queries in the unlinkability game [27].)

- **Initialize**: $\mathcal{R}$ receives a q-GDDH instance

  $$(u_1, u_2, \ldots, u_q, v_1, v_2, \ldots, v_q) \in \mathbb{G}_1^{2q}$$

  sets $(\mathsf{sk}_{\mathsf{Id}}, \mathsf{pk}_{\mathsf{Id}}) \leftarrow \mathsf{IdPKeygen}()$, and sends $\mathsf{pk}_{\mathsf{Id}}$ to $\mathcal{A}$.

- H **Query**: $\mathcal{R}$ forwards this query to Sim, which handles programmability for H.

- H_b **Query**: on receiving query_i, $\mathcal{R}$ checks if it has previously recorded a query-answer pair for this point and sends the same answer if so. Otherwise, $\mathcal{R}$ computes

  $$w_i = (w_{i,1}, w_{i,2}, \ldots, w_{i,q}) \xleftarrow{\$} \mathbb{Z}_p^q$$
  $$b_i \leftarrow \prod_j u_j^{w_{i,j}} \in \mathbb{G}_1$$

  $\mathcal{R}$ records the mapping from query_i to b_i along with auxiliary data w_i, then returns b_i to $\mathcal{A}$.

- **Registration Query**: handled as in the unlinkability game.

- **Ticket Query**: on receiving $(\mathsf{LayerId}_i, \mathsf{EpochId}_i)$, $\mathcal{R}$ queries H_b on $\mathsf{query}_i = \mathsf{LayerId}_i \,||\, \mathsf{EpochId}_i$, then retrieves the auxiliary data w_i associated with query_i and computes

  $$t_i \leftarrow \prod_j v_j^{w_{i,j}} \in \mathbb{G}_1$$

  $\mathcal{R}$ invokes Sim on $\mathsf{LayerId}$, $\mathsf{EpochId}$, and ticket t_i to obtain π_i (recall from above that Sim is able to simulate for a specified ticket in place of a specified NextHop). Finally, $\mathcal{R}$ returns (t_i, π_i) to $\mathcal{A}$.

- **Output**: $\mathcal{R}$ sends $\mathcal{A}$'s output $b^\star$ to the q-GDDH challenger.

$\mathcal{R}$ succeeds with essentially the same probability as $\mathcal{A}$. To see why, first notice that both H and H_b behave (statistically) indistinguishably from random oracles. H_b is immediate: b_i is a random linear combination of $\mathbb{G}_1$ elements and is thus uniformly random in $\mathbb{G}_1$. For H, observe that the points that Sim programs depend on its (secret) random choices, so even with a polynomial number of credentials $\mathcal{A}$ queries a colliding point with negligible probability. Absent such a collision, H's behavior is perfectly simulated.

Next, consider $\mathcal{R}$'s responses to queries. Its responses to registration queries are identical to C's. For ticket queries, first consider

the case that the q-GDDH instance satisfies $\forall j.v_j = u_j^r$; then

$$t_i = \prod_j v_j^{w_{i,j}} = \prod_j \left(u_j^r\right)^{w_{i,j}} = \left(\prod_j u_j^{w_{i,j}}\right)^r = {b_i}^r$$

Since b_i is the basis $H_b(\mathsf{LayerId} \,||\, \mathsf{EpochId})$ and t_i is the ticket, all tickets have discrete log r to their basis, which matches the $b = 0$ case in the unlinkability game (where $r = e + m$). Otherwise, when the v_i in the q-GDDH instance are random group elements, all tickets have a uniformly random discrete-log relation to the corresponding basis. This matches the $b = 1$ case in the unlinkability game because each fresh credential C samples when answering ticket queries induces a random discrete-log relation between ticket and basis.

The above establishes that q-GDDH implies unlinkability. We note that the converse also holds: a set of ticket queries and corresponding H_b evaluations induces a q-GDDH instance whose solution distinguishes $b = 0$ from $b = 1$ in the unlinkability game.

5.3 Availability

We now show that BalancedMixnet provides unforgeability as defined in Section 3.3.2. This property has two components: first, credentials cannot be forged, which holds by the existential unforgeability of the BBS+ signature (§2.2.1). Second, a client can only generate a valid ticket and proof if it holds a valid credential, which holds by the extractability of the proof protocol of Section 4.3.

Finally, we show that BalancedMixnet meets the definition of Sybil Resistance (§3.3.2). First, since tickets are a deterministic function of cred, LayerId, and EpochId, valid tickets are unique and therefore it is not feasible to create multiple valid tickets for a given (cred, LayerId, EpochId) triple. Second, honest servers reject connections during circuit setup if they do not include valid tickets, and messages that do not correspond to valid circuits are dropped. Third, messages are indeed uniformly distributed among a server's outgoing links: for an honest server in the mixnet with ℓ outgoing connections, a message corresponding to ticket t will be sent on connection $j = H(t) \bmod \ell$, where H is a random oracle with codomain $\mathbb{Z}_p = \{0, 1, \ldots, p-1\}$. For security, $p \gg 2^{200}$; meanwhile, $\ell \ll 2^{40}$ (a mixnet with a trillion nodes is ludicrous). $H(t) \bmod \ell$ thus has statistical distance from uniform at most 2^{-160}, which is negligible. (Intuitively: every element in $\{0, \ldots, \ell - 1\}$ is produced by either $\lceil p/\ell \rceil$ or $\lfloor p/\ell \rfloor$ values in the codomain of $H(r)$; these are negligibly different since $p/\ell \gg 2^{160}$.)[4]

6 Evaluation

We implemented a prototype of BalancedMixnet in Rust with 9k lines of code. This includes an implementation of a parallel mixnet and the BalancedMixnet credential and ticketing mechanisms. Our implementation is open-source [35]. It forks the `bbs-crate` library. It extends the validation of the blind signature to also include the NIZK proof for validating BalancedMixnet's tickets (§4.3). Our implementation instantiates the BBS+ signatures over the curve BLS12-381. We use Blake2B to construct the hash functions H and H_b [55], Ed25519 [9] for public-key encryption in the setup-circuit messages, and ChaCha20-Poly1305 [8] for symmetric encryption in the communication rounds. We use our implementation to evaluate BalancedMixnet and its performance impact on parallel mixing. We begin our evaluation by measuring the cryptographic computations BalancedMixnet adds to a mixnet (§6.1). Next, we run system-wide experiments to evaluate BalancedMixnet's end-to-end performance with and without a Sybil attack (§6.2). We compare the impact of integrating BalancedMixnet's load balancing on the performance of two base deployments: a vanilla parallel mixnet [31] and the Yodel mixnet [39].

Yodel is a mixnet-based communication system. In Yodel, each user creates two circuits through a mixnet for communication: a primary circuit, whose address they share with their communication partner so they can connect and read their messages, and a backup circuit, which the user connects to if they fail to communicate their primary circuit's address to their partner. Although Yodel gives a strong privacy guarantee to its users and leverages a circuit-based routing technique to allow low-latency communication, it is susceptible to malicious users who dishonestly select routes, as we show in our evaluation. In Yodel, users are supposed to choose their circuit routes uniformly at random from the full-mesh mixnet topology the system offers (Figure 1a), and message routing over these circuits is synchronous, meaning the slowest server becomes the bottleneck. Extending Yodel with BalancedMixnet's load balancing on top of its already circuit-based communication was simple and required only the addition of calling BalancedMixnet's function for deriving and attaching a ticket when generating circuit setup messages at the client side, and calling BalancedMixnet's ticket verification function when processing them at the mixnet server side.

6.1 Micro-Benchmarks

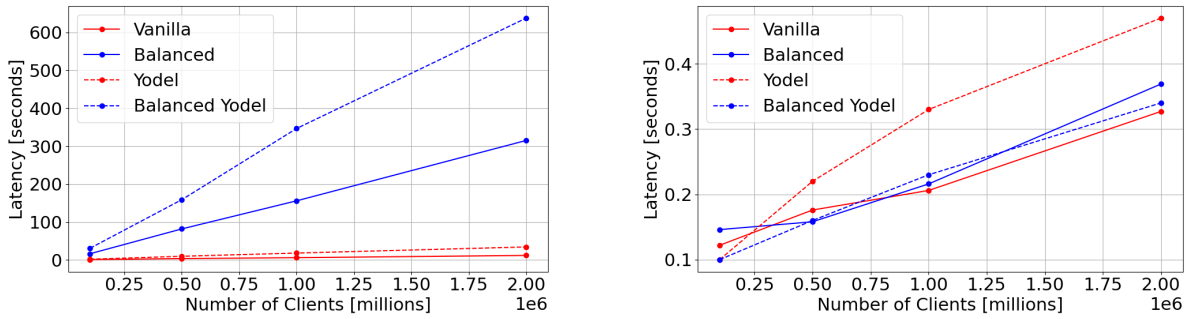
We benchmark the time it takes to run BalancedMixnet's cryptographic computations. For our experiments, we used a Lenovo ThinkPad P16s GEN2 laptop with an Intel I7 1360P core CPU. All the timings reported below are averaged over 100 runs.

Credential issuance time. On registration, the identity provider produces a BBS+ signature used as the client's credential. Each signature takes $352\mu s$ to create. Since registration is not a frequent operation (we envision happening on the order of a month), we believe this would not create a significant performance bottleneck.

Public Key Decryption. At circuit setup, BalancedMixnet mix servers decrypt the public-key-encrypted onion setup messages. We benchmark this decryption to take $120\mu s$. The cost of this decryption is amortized over multiple communication rounds through the lifetime of the circuit.

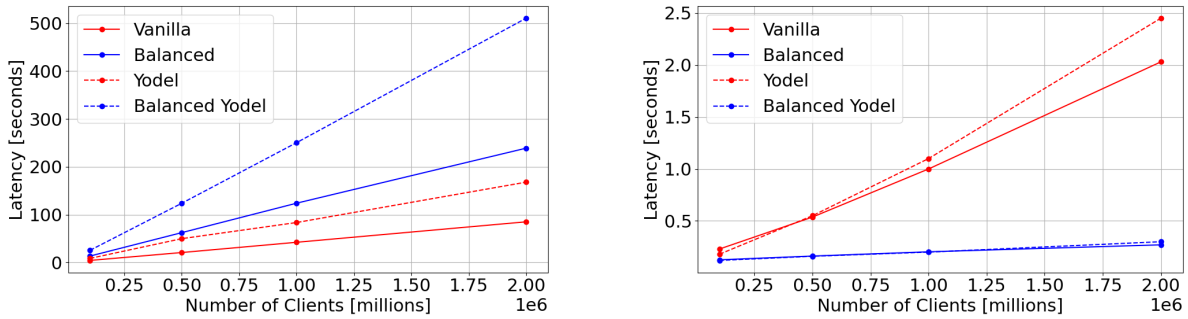
Ticket validation. At circuit setup, each mix server validates a ticket for every message. This constitutes verifying a NIZK proof (§4.3), which we benchmark taking $4.01ms$. Though this is over an order of magnitude more than the standard onion decryption a mix server has to do (using public key cryptography), the ticket validation cost, similarly to the public key decryption, is amortized over multiple communication rounds through the lifetime of the circuit.

[4] If we wanted to sample from a distribution other than uniform, it would suffice to derive a PRG key from t and then use the PRG to run the sampling procedure.



(a) Circuit setup **(b) Communication rounds**

Figure 3: Latency as a function of the user base. None of the clients is malicious. Applying BalancedMixnet's mechanisms adds overhead during circuit setup but not during communication rounds. The overhead impacts similarly when integrating BalancedMixnet on a vanilla mixnet and Yodel.



(a) Circuit setup **(b) Communication rounds**

Figure 4: Latency as a function of the user base. Here, 25% of the clients are malicious and target one middle-layer mix server. BalancedMixnet's communication rounds perform much better than the base vanilla / Yodel mixnets since the load is distributed across all mix servers.

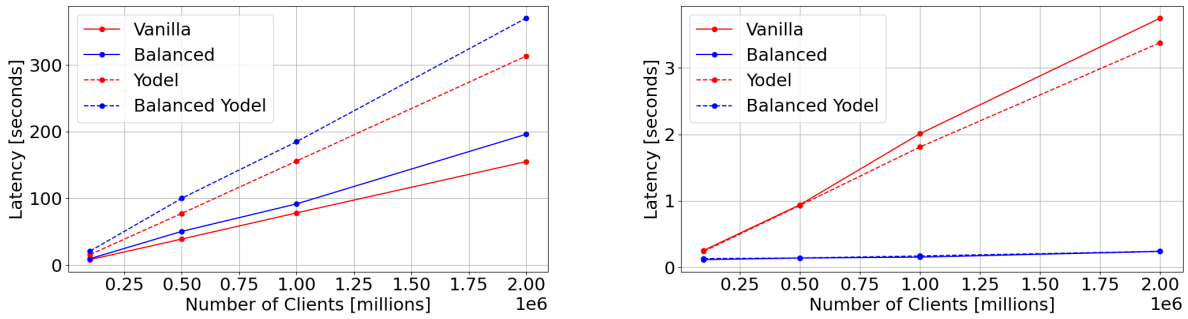
Communication overhead. At circuit setup, BalancedMixnet adds information per mixnet layer. Specifically, it includes a 48B ticket, a 820B NIZK proof the ticket is correct, an 8B ephemeral pseudonym circuit ID, and a 32B symmetric key to onion-encrypt future communication. In total, BalancedMixnet adds 908B per mixnet layer to the circuit setup message. After circuit setup, the communication is very efficient, the data is symmetrically encrypted, and the client only specifies the pseudonym circuit ID for each layer.

6.2 System Performance

We now turn to evaluate how BalancedMixnet impacts performance in the context of a parallel mixnet. Our testbed consists of 80 mix servers deployed in a full mesh topology (Figure 1a) with four layers. Each mix server has four AMD EPYC 7662 $2GHz$ cores.We consider different scenarios of Sybil attacks and compare against the same mixnet deployments, with and without BalancedMixnet's load balancing mechanism. We evaluate two deployments, a vanilla mixnet [31] (solid lines) and the Yodel mixnet [39] (dashed lines).

Our attacker performs a Sybil attack by directing all its clients to route through one mix in each layer. Honest clients, of course, choose routes uniformly for their messages. Therefore, when integrating BalancedMixnet's load balancing, the attacker's circuit setup messages get dropped before the overloaded mix server, while without it, they manage to overload the server. Our evaluation, illustrated in Figures 3 – 5, measures the mixnets' end-to-end latency as a function of the number of clients.

In Figure 3 there is no attack. We observe that BalancedMixnet introduces an overhead in circuit setup, which roughly matches the ratio between doing BalancedMixnet's validation vs. just a public key decryption for the onion message overhead from our microbenchmarks; see Figure 3a. However, circuit setup is an infrequent procedure, which can be performed in the background (i.e., set up the next epoch's circuits before the current epoch is finished, while users communicate over their current circuits). Most of the time is spent sending data messages, and we observe in Figure 3b that there is no performance overhead between the base mixnets and when they integrate BalancedMixnet's load balancing mechanisms.



(a) Circuit setup **(b) Communication rounds**

Figure 5: Latency as a function of the user base. Here, 50% of the clients are malicious and target one middle-layer mix server. BalancedMixnet's overhead during circuit setup becomes very small compared to the vanilla/Yodel mixnets, since it filters the (substantial portion of) malicious clients' messages before a mix server is overloaded. During the communication rounds, the vanilla/Yodel mixnets' performance degrades substantially with the user base, while BalancedMixnet mitigates the attack's impact.

Next, we observe what happens under attack. Consider now an attacker controlling 25% of the clients (Figure 4). In this case, we still see that BalancedMixnet's circuit setup takes longer than the base mixnets, due to the cost of validating the clients' tickets. However, BalancedMixnet effectively mitigates the impact of the Sybil attacker when users start communicating over the mixnet. We observe that the latency of the base mixnets increases substantially with the user base. As the number of rogue clients grows, so does the load on the target mix server and, therefore, the slower the base mixnets become. When the mixnets integrate BalancedMixnet, on the other hand, they are not impacted by the attacker. In this case, the communication latency grows slowly with the number of clients, which is to be expected, since the processing load increases proportionally on all mixes (as intended by the parallel mixnet design).

Lastly, we consider an attacker controlling 50% of the clients. In this scenario, we observe that BalancedMixnet's overhead during circuit setup is small compared to the base mixnets. This is because in the base mixnets, one server becomes the bottleneck of decrypting the circuit setup messages, while BalancedMixnet filters those rogue messages across all mixes in the preceding layer, which avoids the bottleneck and counters the impact of the computational expense of validating tickets. The impact of BalancedMixnet's defense against Sybil attacks is apparent during the communication rounds since, in the base mixnets, one of the layers includes a server overloaded by handling most of the traffic, which slows down the whole synchronous system. In contrast, the user load when integrating BalancedMixnet is uniformly distributed across all servers, which mitigates the attack in both the vanilla and Yodel deployments.

7 Related Work

This is the first work, to our knowledge, to ensure proper load balancing of a mixnet's user load while maintaining user anonymity. There has been significant work studying and mitigating Sybil attacks in other contexts, mostly where Sybil attacks directly compromise security. For example, addressing such attacks is at the core of blockchain security [48], where a consensus protocol requires an honest majority. It was also studied extensively in the context of ad hoc networks where Sybil participants can mislead network-wide decisions [17, 34].

In the context of privacy, anonymizing user messages by mixing them in fixed-size batches is known to be sensitive to Sybil attacks if the attacker can infiltrate many of their own messages into such a batch. This was pointed out specifically for parallel mixnets [13], as well as other contexts like mixing anonymous payments [10, 45]. The solution is typically to avoid fixed-size mixing and instead shuffle all messages arriving in a time interval. In the context of parallel mixnets, this allows the attacker to launch the class of Sybil attacks tackled by this work (namely, disproportionally overloading a particular mix server).

Most prior work dealing with mixnet Sybil attacks focused on handling colluding mix servers [21, 40]. That is, a case where it is challenging for clients to select a route with enough honest servers to anonymize their messages. BalancedMixnet, in contrast, focuses on resisting Sybil clients and ensuring uniform random load distribution across servers while keeping user privacy. Using random assignment for load balancing was also used in other contexts pertaining to privacy, for example, Chow et al. [19] use it to ensure a sustainability property in systems that require users to keep creating new pseudonyms (e.g., wallet addresses/unspent transaction outputs in anonymous payment systems).

Using anonymous credentials. Restricting credential transfer was one of the design goals of anonymous credentials. The original design bounded the credential given to the user to their private key, so transferring it from one user to another required revealing this secret, which served to discourage such transfers [16]. Techniques for ensuring that anonymous credentials cannot be used more than once were later presented [7], and framed anonymous credentials as means for mitigating Sybil attacks [41].

BLAC [53], like BalancedMixnet, uses BBS+ signature-based anonymous credentials, but in a different context. It shows how to

implement a blacklist of users although they have an anonymous credential. In particular, clients derive access tickets from their credential like in BalancedMixnet, and they include for each of their past tickets a zero knowledge proof that it wasn't put on the blacklist. While the anonymous credential technique is similar, using BLAC's method for leveraging anonymous credentials to block misbehaving users is not suitable for our parallel mixnet use case, since it would not scale well with the number of users (which is the primary goal of using parallel mixing). In particular, even if the attacking clients were blocked, BLAC requires everyone, including all honest clients, to sync with the blacklist from some server and produce long proofs (linear in the size of the blacklist) showing that their current tickets are not blacklisted. The verification time for those proofs is also linear in the blacklist size, which would lead to continuous degradation in performance even after the attack stops. Lastly, the BBS+ signatures used in BLAC [3, 4] don't work with type 3 pairings, in contrast to the scheme used in this work [14]; this forces differences in the protocol for generating tickets and results in BalancedMixnet's proofs for validating tickets being more efficient.[5]

Chow formalized an alternative approach to BalancedMixnet for issuing privacy-preserving credentials to users admitted to a system [20]. Under this approach, the identity provider is split into two entities (that are assumed to be non-colluding), one that issues a user a commitment to their real identity and another that issues the credential based on this commitment, without seeing the users' identities. This approach may allow implementing a wide range of Sybil-resistant admission control policies since users reveal their identity to the first server (extending the privacy-friendly examples we bring up in §3.1). BalancedMixnet can integrate it by replacing its admission approach, which avoids this revelation; however, this approach may be less appealing to privacy-caring users who wish to keep their identities completely hidden.

Achieving Sybil-resistance through trusted randomness as an alternative approach. An alternative baseline design to BalancedMixnet's reliance on credentials may be using beacons that provide a trusted form of randomness [49]. Under this approach, clients register a public key with the system, and the beacon publishes a (trusted) random value every epoch. The clients then derive the route by computing a verifiable random function using their secret key and the beacon's random value. To avoid linking their public key to the route they use, clients would need to attach a NIZK proof per hop proving that the route they derived is valid while hiding their key. This design is certainly possible, but we believe that for a mixnet system, BalancedMixnet's design is simpler and more self-contained since it does not rely on an external beacon (often implemented through a complex system, like a blockchain [29] or hierarchical publicly verifiable secret sharing [49]), which may rely on additional assumptions for liveness and security; e.g., requiring that a majority of a cryptocurrency's tokens are in honest hands [29].

[5]We acknowledge that modifying BLAC to use type-3 pairings is likely possible, and we expect some of the differences in the ticket-generating protocol would carry over. However, evaluating such a change is beyond the scope of this comparison.

Another variant of a trusted randomness approach is having servers shuffle messages and assign them to the next server according to the trusted randomness (and the server's secret key) and provide a NIZK proof of correct shuffle to the other servers, extending standard verifiable shuffle proofs [33, 42]. Under this design, the mixnet uses in-network routing rather than the standard source routing technique. This departure introduces significant challenges. First, a server's proof must be checked by other servers (e.g., other mix servers), or the corrupt server could just choose to relay Alice's message to another corrupt server, which introduces overhead (e.g., the verifiers must have information about the messages a mix receives and outputs to verify the shuffle, and this holds for every mix server, limiting scalability). Furthermore, since an attacker may have some control over a corrupt mix server's input message batch, it may manipulate it such that Alice's message gets routed to another corrupt server (even if the shuffle is verified).

Systems where admission control is undesired or can be centralized. BalancedMixnet mitigates Sybil attacks by introducing admission control, but some systems may be unable to introduce such a mechanism because of the nature of the problem they solve. For example, the Algorand blockchain [29] is designed to be permissionless (without admission control), yet for security, it requires running a lottery across its users where the winner remains anonymous for some time span. For such systems, implementing a trusted source of randomness (as mentioned above) may be a better approach. In the particular example above, Algorand exploits its blockchain application to implement a trusted source of randomness by having block proposers perturb the current random seed using a verifiable random function. Each user computes their priority for proposing the next block using their public key and the current random seed in a process termed cryptographic sortition. When honest users win the lottery (have the highest priority) and propose a block, they are, in essence, randomizing the seed to make it unpredictable, and the blockchain consensus ensures everyone knows that seed. This method is very resilient to churn, since even if many users are offline, some user would win the lottery, randomize the seed, and allow the system to make progress. To ensure that honest users win the lottery often enough and that the consensus protocol is sound, Algorand must make an assumption about the distribution of currency honest users hold, an extra assumption that BalancedMixnet does not need.

Other types of systems where privacy is crucial may require admission control but can rely on simpler mechanisms than BalancedMixnet's anonymous credential-based mechanism. For example, PIR-based systems let users read data from a server without revealing to the server which piece of information they were reading. PIR was also used in the context of metadata-private communication, where users write messages to the server and send PIR queries to the server to read a message from their friends [1, 2, 52]. For PIR-based systems, the server-side cost for processing users' queries is much higher than processing their messages in a mixnet. However, despite the high processing costs, such systems do not face the same Sybil attack challenge for dealing with rogue users as mixnet-based systems. The reason is that users typically contact a centralized PIR server directly to read messages since the server processing the query anyway learns nothing about which data they were reading

(PIR's threat model assumes that servers might be completely malicious, whereas mixnets require that there are some honest servers). Therefore, if a user floods the server with many queries, the PIR server can just block them. For such systems, a simple mechanism for user credentials, such as access tokens that clients attach to messages and queries they submit, may be sufficient.

8 Conclusion

We presented BalancedMixnet, a Sybil resistance mechanism that ensures clients are well-distributed across a parallel mixnet's servers while maintaining their anonymity. BalancedMixnet achieves load balancing across servers by introducing admission control to the mixnet, which issues clients anonymous credentials. The admission policy should preserve user privacy while simultaneously resisting Sybil attacks; we outline several examples for such policies. Clients then use their credentials to create tickets that prove they route their messages correctly, which also ensures their messages are distributed uniformly across the mixnet's servers. An evaluation of a prototype implementation shows that the cost of adopting BalancedMixnet is small, and its benefit against Sybil attacks is substantial. We believe this makes BalancedMixnet an important component in deploying parallel mixnets in practice, where clients can misbehave and attackers may try to exploit this to cripple performance and cause users to leave the system in favor of a non-private communication medium.

Acknowledgments

We thank the anonymous reviewers and shepherd for their comments, suggestions and feedback, which have helped improve this paper. This work was supported by Israel Science Foundation grant no. 1104/23.

References

[1] Ishtiyaque Ahmad, Yuntian Yang, Divyakant Agrawal, Amr El Abbadi, and Trinabh Gupta. Addra: Metadata-private voice communication over fully untrusted infrastructure. *SOSP*, 2021.
[2] Sebastian Angel and Srinath T. V. Setty. Unobservable communication over fully untrusted infrastructure. In Kimberly Keeton and Timothy Roscoe, editors, *OSDI*, pages 551–569. USENIX Association, 2016.
[3] Man Ho Au, Willy Susilo, and Yi Mu. Constant-size dynamic k-TAA. In *SCN*, September 2006.
[4] Man Ho Au, Willy Susilo, Yi Mu, and Sherman S. M. Chow. Constant-size dynamic k -times anonymous authentication. *IEEE Syst. J.*, 7(2):249–261, 2013.
[5] Feng Bao, Robert H. Deng, and Huafei Zhu. Variations of Diffie-Hellman problem. In *ICICS*, October 2003.
[6] Ludovic Barman, Moshe Kol, David Lazar, Yossi Gilad, and Nickolai Zeldovich. Groove: Flexible metadata-private messaging. In *OSDI*, July 2022.
[7] Mira Belenkiy, Melissa Chase, Markulf Kohlweiss, and Anna Lysyanskaya. P-signatures and noninteractive anonymous credentials. In *Theory of Cryptography: Fifth Theory of Cryptography Conference, TCC 2008, New York, USA, March 19-21, 2008. Proceedings 5*, pages 356–374. Springer, 2008.
[8] Daniel J. Bernstein. ChaCha: A variant of Salsa20. In *SASC*, January 2008.
[9] Daniel J. Bernstein, Niels Duif, Tanja Lange, Peter Schwabe, and Bo-Yin Yang. High-speed high-security signatures. In *CHES*, October 2011.
[10] George Bissias, A Pinar Ozisik, Brian N Levine, and Marc Liberatore. Sybil-resistant mixing for bitcoin. In *WPES*, October 2014.
[11] Dan Boneh and Xavier Boyen. Short signatures without random oracles and the SDH assumption in bilinear groups. *J Cryptology*, 21:149–177, 2008.
[12] Maria Borge, Eleftherios Kokoris-Kogias, Philipp Jovanovic, Linus Gasser, Nicolas Gailly, and Bryan Ford. Proof-of-personhood: Redemocratizing permissionless cryptocurrencies. In *2017 IEEE European Symposium on Security and Privacy Workshops (EuroS&PW)*, pages 23–26. IEEE, 2017.
[13] Nikita Borisov. An analysis of parallel mixing with attacker-controlled inputs. In *PET*, May 2005.
[14] Jan Camenisch, Manu Drijvers, and Anja Lehmann. Anonymous attestation using the strong diffie hellman assumption revisited. In *TRUST*, August 2016.
[15] Jan Camenisch, Aggelos Kiayias, and Moti Yung. On the portability of generalized schnorr proofs. In *Eurocrypt*, April 2009.
[16] Jan Camenisch and Anna Lysyanskaya. An efficient system for non-transferable anonymous credentials with optional anonymity revocation. In *International conference on the theory and applications of cryptographic techniques*, pages 93–118. Springer, 2001.
[17] Wei Chang and Jie Wu. A survey of sybil attacks in networks. *Sensor networks for sustainable development*, 2012.
[18] David L. Chaum. Untraceable electronic mail, return addresses, and digital pseudonyms. *Commun. ACM*, 24(2), February 1981.
[19] Sherman SM Chow, Christoph Egger, Russell WF Lai, Viktoria Ronge, and Ivy KY Woo. On sustainable ring-based anonymous systems. In *2023 IEEE 36th Computer Security Foundations Symposium (CSF)*, pages 568–583. IEEE, 2023.
[20] Sze-Ming Chow. *New privacy-preserving architectures for identity-/attribute-based encryption.* PhD thesis, New York University, 2010.
[21] George Danezis, Chris Lesniewski-Laas, M Frans Kaashoek, and Ross Anderson. Sybil-resistant DHT routing. In *ESORICS*, September 2005.
[22] Roger Dingledine and Nick Mathewson. Anonymity loves company: Usability and the network effect. In *WEIS*, 2006.
[23] Roger Dingledine, Nick Mathewson, and Paul F. Syverson. Tor: The second-generation onion router. In *USENIX Security*, August 2004.
[24] Roger Dingledine, Vitaly Shmatikov, and Paul Syverson. Synchronous batching: From cascades to free routes. In *International Workshop on Privacy Enhancing Technologies*, pages 186–206. Springer, 2004.
[25] Jack Doerner, Yashvanth Kondi, Eysa Lee, Abhi Shelat, and LaKyah Tyner. Threshold BBS+ signatures for distributed anonymous credential issuance. In *IEEE S&P*, May 2023.
[26] Sebastian Faust, Carmit Hazay, David Kretzler, Leandro Rometsch, and Benjamin Schlosser. Non-interactive threshold bbs+ from pseudorandom correlations. In *Cryptographers' Track at the RSA Conference*, pages 198–222. Springer, 2025.
[27] Marc Fischlin, Anja Lehmann, Thomas Ristenpart, Thomas Shrimpton, Martijn Stam, and Stefano Tessaro. Random oracles with(out) programmability. In *ASIACRYPT*, December 2010.
[28] Steven D. Galbraith, Kenneth G. Paterson, and Nigel P. Smart. Pairings for cryptographers. *IACR Cryptol. ePrint Arch.*, page 165, 2006.
[29] Yossi Gilad, Rotem Hemo, Silvio Micali, Georgios Vlachos, and Nickolai Zeldovich. Algorand: Scaling byzantine agreements for cryptocurrencies. In *Proceedings of the 26th symposium on operating systems principles*, pages 51–68, 2017.
[30] gk. Malicious relays and the health of the Tor network. https://blog.torproject.org/malicious-relays-health-tor-network/, 2022.
[31] Philippe Golle and Ari Juels. Dining cryptographers revisited. In Christian Cachin and Jan Camenisch, editors, *EUROCRYPT*, volume 3027 of *Lecture Notes in Computer Science*, pages 456–473. Springer, 2004.

[32] Philippe Golle and Ari Juels. Parallel mixing. In *ACM CCS*, October 2004.
[33] Jens Groth and Steve Lu. Verifiable shuffle of large size ciphertexts. In *Public Key Cryptography–PKC 2007: 10th International Conference on Practice and Theory in Public-Key Cryptography Beijing, China, April 16-20, 2007. Proceedings 10*, pages 377–392. Springer, 2007.
[34] Rasheed Hussain and Heekuck Oh. On secure and privacy-aware sybil attack detection in vehicular communications. *Wireless Personal Communications*, 77:2649–2673, 2014.
[35] Maya Kleinstein. BalancedMixnet prototype implementation. https://github.com/maya-kleinstein/Sybil-Resistant-Mixnet/tags, 2025.
[36] Albert Kwon, Henry Corrigan-Gibbs, Srinivas Devadas, and Bryan Ford. Atom: Horizontally scaling strong anonymity. In *SOSP*, October 2017.
[37] Albert Kwon, David Lu, and Srinivas Devadas. XRD: Scalable messaging system with cryptographic privacy. In *NSDI*, February 2020.
[38] David Lazar, Yossi Gilad, and Nickolai Zeldovich. Karaoke: Distributed Private Messaging Immune to Passive Traffic Analysis. In *OSDI*, September 2018.
[39] David Lazar, Yossi Gilad, and Nickolai Zeldovich. Yodel: Strong metadata security for voice calls. In *SOSP*, October 2019.
[40] Hemi Leibowitz, Ania M Piotrowska, George Danezis, and Amir Herzberg. No right to remain silent: isolating malicious mixes. In *USENIX Security*, August 2019.
[41] Leonardo A Martucci, Markulf Kohlweiss, Christer Andersson, and Andriy Panchenko. Self-certified sybil-free pseudonyms. In *Proceedings of the first ACM conference on Wireless network security*, pages 154–159, 2008.
[42] C Andrew Neff. A verifiable secret shuffle and its application to e-voting. In *Proceedings of the 8th ACM conference on Computer and Communications Security*, pages 116–125, 2001.
[43] Torben Pryds Pedersen. Non-interactive and information-theoretic secure verifiable secret sharing. In *CRYPTO*, August 1991.
[44] Ania M. Piotrowska, Jamie Hayes, Tariq Elahi, Sebastian Meiser, and George Danezis. The Loopix anonymity system. In *USENIX Security*, August 2017.
[45] Mikerah Quintyne-Collins. Short paper: Towards characterizing sybil attacks in cryptocurrency mixers. Cryptology ePrint Archive, Report 2019/1111.
[46] Rarimo: ZK Passport. https://docs.rarimo.com/zk-passport/.
[47] Claus-Peter Schnorr. Efficient signature generation by smart cards. *J Cryptology*, 4(3):161–174, 1991.
[48] P Swathi, Chirag Modi, and Dhiren Patel. Preventing sybil attack in blockchain using distributed behavior monitoring of miners. In *ICCCNT*, July 2019.
[49] Ewa Syta, Philipp Jovanovic, Eleftherios Kokoris Kogias, Nicolas Gailly, Linus Gasser, Ismail Khoffi, Michael J Fischer, and Bryan Ford. Scalable bias-resistant distributed randomness. In *2017 IEEE Symposium on Security and Privacy (SP)*, pages 444–460. Ieee, 2017.
[50] The Self team. The Self Project. https://github.com/selfxyz/self, 2025.
[51] The NYM team. Nym. https://nymtech.net/, 2022.
[52] Elkana Tovey, Jonathan Weiss, and Yossi Gilad. Distributed pir: Scaling private messaging via the users' machines. In *Proceedings of the 2024 on ACM SIGSAC Conference on Computer and Communications Security*, pages 1967–1981, 2024.
[53] Patrick P Tsang, Man Ho Au, Apu Kapadia, and Sean W Smith. BLAC: Revoking repeatedly misbehaving anonymous users without relying on TTPs. *ACM Trans Information and System Security*, 13(4):1–33, 2010.
[54] Nirvan Tyagi, Yossi Gilad, Derek Leung, Matei Zaharia, and Nickolai Zeldovich. Stadium: A distributed metadata-private messaging system. In *SOSP*, October 2017.
[55] Riad S. Wahby and Dan Boneh. Fast and simple constant-time hashing to the BLS12-381 elliptic curve. In *CHES*, August 2019.
[56] Hoeteck Wee. Zero knowledge in the random oracle model, revisited. In *ASIACRYPT*, December 2009.
[57] Harry WH Wong, Jack PK Ma, and Sherman SM Chow. Secure multiparty computation of threshold signatures made more efficient. In *ISOC Network and Distributed System Security Symposium–NDSS 2024*, 2024.