# Arke: Scalable and Byzantine Fault Tolerant Privacy-Preserving Contact Discovery

Nicolas Mohnblatt
nico@geometry.dev
Geometry Research
Netanya, Israel

Alberto Sonnino
alberto@mystenlabs.com
Mysten Labs
London, United Kingdom
University College London
London, United Kingdom

Kobi Gurkan
kobi@geometry.dev
Geometry Research
Netanya, Israel

Philipp Jovanovic
p.jovanovic@ucl.ac.uk
University College London
London, United Kingdom

## Abstract

Contact discovery is a crucial component of social applications, facilitating interactions between registered contacts. This work introduces Arke, a novel contact discovery scheme that addresses the limitations of existing solutions in terms of privacy, scalability, and reliance on trusted third parties. Arke ensures the unlinkability of user interactions, mitigates enumeration attacks, and operates without single points of failure or trust. Notably, Arke is the first contact discovery system whose performance is independent of the total number of users and the first that can operate in a Byzantine setting. It achieves its privacy goals through an unlinkable handshake mechanism built on top of an identity-based non-interactive key exchange. By leveraging a custom distributed architecture, Arke forgoes the expense of consensus to achieve scalability while maintaining consistency in an adversarial environment. Performance evaluations demonstrate that Arke provides a throughput of over 1,500 user requests per second at a latency of less than 0.5 seconds in a large geo-distributed setting which would allow privacy-preserving contact discovery for all of the popular messaging applications in one system.

## CCS Concepts

• **Security and privacy → Social network security and privacy**; **Privacy-preserving protocols**.

## Keywords

Private Contact Discovery; Blockchain; Identity-based Key Exchange; BFT

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than the author(s) must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.
*CCS '24, October 14–18, 2024, Salt Lake City, UT, USA*
© 2024 Copyright held by the owner/author(s). Publication rights licensed to ACM.
ACM ISBN 979-8-4007-0636-3/24/10
https://doi.org/10.1145/3658644.3670289

**ACM Reference Format:**
Nicolas Mohnblatt, Alberto Sonnino, Kobi Gurkan, and Philipp Jovanovic. 2024. Arke: Scalable and Byzantine Fault Tolerant Privacy-Preserving Contact Discovery. In *Proceedings of the 2024 ACM SIGSAC Conference on Computer and Communications Security (CCS '24), October 14–18, 2024, Salt Lake City, UT, USA.* ACM, New York, NY, USA, 15 pages. https://doi.org/10.1145/3658644.3670289

## 1 Introduction

Contact discovery enables users of social applications, such as messengers, payment systems, or media-sharing platforms, to find and interact with their registered contacts [55]. Consider for example Signal [72]: users sign up with their phone numbers, and want to connect with any phone number of their address book that is also registered on Signal. This process can be generalized to allow users to sign up with any form of identity (a blockchain address, private-public key pair) and be found using any convenient human-readable identifier (email, phone number, social media handle).

Current solutions have significant shortcomings in meeting several important expectations. Some fail to adequately protect users' privacy, exposing their underlying social relations either by design [75] or when targeted by enumeration or crawling attacks [44]. These solutions often rely on centralized parties [26] or trusted hardware for privacy protection [56]. Finally, all these solutions express some form of dependency on the total number of users (either in latency, computation or storage) and may not be suitable for applications with billions of users[1].

Arke[2] is a novel contact discovery scheme that addresses the limitations found in existing systems. Arke ensures the unlinkability of user interactions and effectively mitigates enumeration attacks. It prioritizes user privacy by ensuring that no information about users, their messages, or their communication partners is revealed. Additionally, Arke enforces a bi-directional relationship requirement, meaning that users can only discover each other if they are mutually seeking contact. This approach prevents crawling attacks, setting it apart from traditional contact discovery schemes. Furthermore, Arke supports multiple applications sharing the same

[1] WhatsApp, the most popular end-to-end encrypted messaging application, was reported to have 2.7 billion unique active users in June 2023 [23].
[2] In Greek mythology, Arke is the messenger of the Titans.

contact discovery infrastructure while maintaining independent security assumptions. Notably, Arke represents a significant advancement as the first privacy-preserving contact discovery system whose performance is independent of the total number of users in the system (often referred to as the database size). Moreover, Arke stands out as the first contact discovery system designed without any single points of failure or trust; Arke offers scalability in terms of throughput and extremely low latency despite the presence of a Byzantine adversary.

The Arke contact discovery protocol generalizes the construction of Chaum *et al.* [26], known as *UDM* (User Discovery with Minimal information disclosure). Implicit to the UDM architecture is the fact that a contact discovery scheme can be built by combining a *key exchange* and an *unlinkable handshake* [47]. First, users run a key exchange to establish a shared secret. Then, using this secret, the users run the handshake protocol to establish an end-to-end encrypted channel, without revealing any connection details to third parties. Finally, the channel is used to exchange any information that is needed to interact with each other on the new social application. Chaum *et al.* [26] realize both of these subprotocols with the help of centralized parties (the *Public-Key Manager* and *Encrypted ID Manager* respectively). Arke improves on these requirements. The key exchange is instantiated with a variant of the Sakai-Ohgishi-Kasahara identity-based non-interactive key exchange (ID-NIKE) [71]. By utilizing distributed key generation [41] and blind threshold BLS signatures [15], we modify the original protocol to distribute the master secret key and enable oblivious and verifiable key issuance. We then present a custom unlinkable handshake protocol which only requires an untrusted (and potentially distributed) public bulletin board. The design of this handshake ensures that each system resource is mutated by at most a single user, eliminating the need for an expensive consensus protocol to maintain consistency in the distributed setting. Instead, Arke relies on a simpler and more efficient primitive based on Consistent Broadcast [20].

By construction, Arke supports applications beyond contact discovery. Indeed, the protocol facilitates a rendezvous point and allows the exchange of an arbitrary message in a privacy-preserving manner. This message may include public keys, in view of a key exchange for a forward-secure E2EE messenger, or transaction authorizations. The latter allows users to pay contacts directly, knowing only their non-cryptographic identifiers, even before the contacts have generated a wallet or account in the relevant payment system. This provides a convenient mechanism for user onboarding or airdrops, for example.

We implement and evaluate a prototype of Arke written in Rust on Amazon EC2 in a large geo-distributed wide-area network deployment. We show that after a short one-time offline phase taking only a couple of seconds, Arke supports over 1,500 user requests per second with a latency of less than 0.5 seconds even when the infrastructure is maintained by 50 authorities. Furthermore, Arke can maintain this throughput with sub-second latency even when up to a third of these authorities fail.

*Contributions.* This paper makes the following contributions:

- It presents Arke, a novel privacy-preserving contact-discovery construction that is the first with performance independent of the total number of users in the system, and the first designed to operate in a Byzantine environment. It does so by generalizing UDM [26] and by introducing a threshold and oblivious variant of the Sakai-Ohgishi-Kasahara ID-NIKE [71], as well as a custom unlinkable handshake.
- It proves the security and privacy guarantees of the system (left as open question in Chaum *et al.* [26]).
- It shows how Arke maintains consistency of a distributed key-value store without requiring consensus but instead using simpler and more efficient broadcast-based primitives.
- It provides a full implementation of Arke and a performance evaluation on a real geo-distributed environment under varying system loads and fault scenarios.
- It shows how existing blockchains can leverage Arke to build a privacy-preserving contact discovery service for their wallets, and how messaging services such as Signal [72], Telegram [1], and WhatsApp [79] can run Arke to allow users to privately discover each other's public keys.

## 2 System Overview

Arke enables Alice to discover a *message* $\mathsf{msg}_B$ from a sender Bob known only by his *identifier* $\mathsf{id}_B$ through the establishment of a shared cryptographic secret between them. An identifier is a public human-readable string unique to a user, such as a phone number, an email address, or a social media handle. Arke is efficient and privacy-friendly by hiding the identifiers, messages, and relationships between users.

### 2.1 Actors

Arke is composed of the following actors.

*Users.* A *user*, Alice, owns a human-readable identifier $\mathsf{id}_A$ and a message (or payload) $\mathsf{msg}_A$. She wishes to allow specific users to discover her message on the conditions that (i) Alice knows the other user's identifier and (ii) the other user knows Alice's identifier. Users wish to hide their relationships with other users from any observer.

*Registration Authorities.* A *registration authority* (RA) attests to the binding between users and their identifiers. A registration authority could be a social media service (e.g., X - formerly known as Twitter) allowing the use of usernames as identifiers or a messaging service verifying a phone number, or any third party running an interactive protocol with the user to verify their identifiers (e.g., by sending them a text code, or running a protocol to authenticate TLS data [24, 25, 54, 81, 83]). Identifiers always specify the registration authority that attested to them. As a result, multiple services (e.g., Signal [72], Telegram [1], WhatsApp [79], or any third-party service) can all use the user's phone number as an identifier by appending their unique RA domain, e.g., `phone_number@domain`. A registration authority can be a single entity or a distributed set of authorities. The concrete deployment structure is decided by the respective service designers/operators. For simplicity of presentation, we assume henceforth that a registration authority is a single entity.

*Key-issuing Authorities.* The *key-issuing authorities* (KAs) are a committee of $n$ entities that share a threshold key (see Section 4).

They are tasked with issuing private keys to users who present a valid proof of registration. Arke assumes that at most $t < n/2$ key-issuing authorities are Byzantine (see Section 2.4).

*Storage Authorities.* The Arke storage is operated by a set of $3f + 1$ independent *storage authorities* out of which at most $f$ are Byzantine (see Section 2.3). We present the storage authorities as independent entities but they may coincide with the key-issuing authorities (by setting $t = f$) or coincide with the maintainers of most existing blockchains (see Section 5.2). In the general setting, storage authorities may enforce their own access control policy and only accept write requests from users registered with RAs of their choice.

## 2.2 Protocol Outline

Arke is divided into two phases: *(i)* a *setup phase* where users obtain a long-term private key over their identifier, and *(ii)* a *discovery phase* where users use their private keys to anonymously exchange messages with their contacts over an untrusted public message board. The setup phase is executed only once (or rarely) and the discovery phase is executed every time a user wishes to make her message discoverable or discover the message of a contact. Figure 1 provides an overview of Arke and the interactions between its actors.

*Setup phase.* Alice convinces a registration authority that she owns the identifier $\mathrm{id}_A$ and receives a signed attestation in return (❶). She then blinds her identifier and attestation to submit anonymous key-issuance requests to at least $t+1$ key-issuing authorities. Upon verifying a request, each key-issuing authority blindly emits a share of Alice's private key. Finally, Alice locally combines the shares to obtain her long-term private key (❷).

*Discovery phase.* After running the setup phase, Alice wishes to signal to Bob that she has registered and optionally sends him a message. Using her long-term private key and Bob's identifier, Alice locally derives a shared secret with Bob (❸). From this shared secret, Alice can derive a label and a symmetric key used for encryption. She encrypts her message and writes the ciphertext and label to the distributed Arke store (❹). Bob can discover Alice's message by locally deriving the same shared secret (using his long-term private key and Alice's identifier) (❺) and reading the distributed Arke store (❻). Arke divides time in a sequence of epochs (e.g., lasting about 1 or 2 weeks). After a fixed number of epochs, the storage authorities delete the records of inactive users. This garbage-collection mechanism is detailed in the extended version of the paper[3].

## 2.3 Design Goals

Arke guarantees several system security, privacy, and performance properties.

*System security properties.* Arke maintains several systems security properties depending on which assumptions (Section 2.4) hold.

- **Validity:** Alice can only update the Arke store by updating messages associated with her identifier $\mathrm{id}_A$.
- **Write consistency:** No correct storage authorities hold conflicting records.
- **Read consistency:** No two read operations over the same label return a different ciphertext.
- **Write termination:** A correct user can eventually update the store to make its message discoverable.
- **Read termination:** A correct user can eventually read the store and learn the message associated with a user with a known identifier.

*Privacy properties.* Arke upholds the following properties:

- **Anonymity**: The identities of active Arke users are kept hidden from the key-issuing authorities, storage authorities, and any third-party observer. Identities may also be hidden from the relevant registration authority if their authentication mechanism is anonymous. This mechanism is left at the discretion of each registration authority and is out of our design scope.
- **Confidentiality**: Messages exchanged over Arke are encrypted and recipient-anonymous.
- **Unlinkability**: None of the authorities or third-party observers can determine whether Alice and Bob have exchanged messages over Arke.
- **Selective discovery**: Users may choose whether or not to be discoverable by other users on a *per-user* basis. The default behavior is to remain hidden. This property contrasts with other contact discovery schemes where users make themselves discoverable to all, allowing crawling attacks as studied by Hagen *et al.* [44].

*Performance properties.* Arke also guarantees the following system and performance properties. Section 6 demonstrates these properties through a thorough implementation and evaluation of Arke.

- **High-throughput:** Arke provides enough throughput to support multiple applications with billions of users each; we estimate that Arke can support the combined user base of WhatsApp, Facebook Messenger, Signal, and Telegram.
- **Low-latency:** Arke achieves sub-second latency even for large geo-distributed deployments.
- **Performance under (crash-)faults:** The performance (throughput and latency) of Arke is virtually unaffected by (crash-)faulty authorities. Note that evaluating a BFT system while experiencing Byzantine faults is an open problem [12].
- **Bounded storage**: Storage is not growing linearly over time. Arke enables authorities to periodically purge their store entries. This property is proven as part of *consistency*.

*Additional properties.* Furthermore, Arke guarantees the following meta-properties:

- **Censorship resistance**: Correct users can always obtain private keys from the key-issuing authorities. Furthermore, correct users can write and read the Arke store despite the presence of Byzantine authorities. This property is proved as part of *write termination* and *read termination*.
- **Authorities Non-Interactivity**: Neither the Arke key-issuing authorities nor the storage authorities need to communicate with each other. This property allows for easier deployment and is crucial to integrate Arke into the Sui blockchain [57] (see Section 5.2).

[3]The extended version of the paper is available at https://eprint.iacr.org/2023/1218

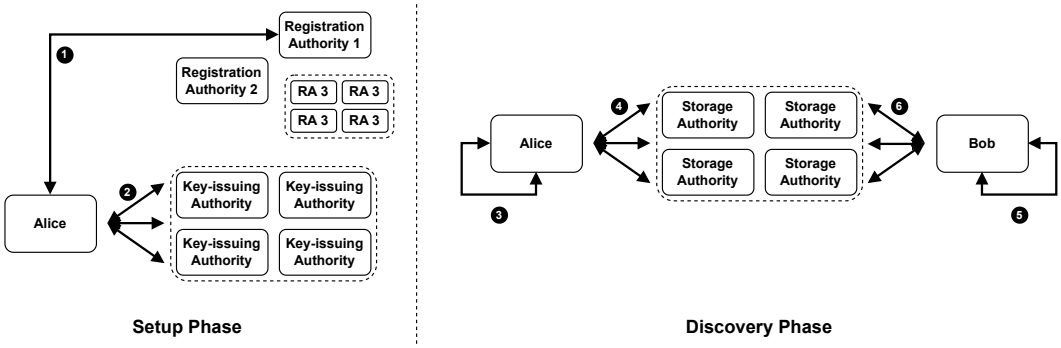


**Figure 1:** *Arke* **overview. During the setup phase, users run an (anonymous) identification procedure with a registration authority to obtain an attestation over their identifier (❶). They then use this attestation along with their blinded identifier to obtain a long-term private key by interacting with the key-issuing authorities (❷). During the discovery phase, users locally derive a shared secret with each contact (❸,❺) and use it to read and write the Arke distributed store and discover their messages (❹,❻).**

## 2.4 Threat Model

We define the main assumptions under which Arke guarantees the properties of Section 2.3.

**Assumption 1: Correct registration authorities.** Arke guarantees the security properties of Section 2.3 for identifiers attested by correct registration authorities. Indeed, a malicious RA could falsely issue attestations and impersonate any user it desires. Fortunately, recent work on authenticating web data has shown that privacy-preserving, untrusted and correct RAs can be realized in practice [24, 25, 54, 81, 83]. Some of these solutions are under active development at the time of publication of this work [76]. Additionally, Arke mitigates the threat of malicious RAs by confining each RA to a unique domain (see Section 2.1 and Section 4.3).

**Assumption 2: BFT authorities.** Arke assumes a computationally bounded adversary that controls the network and can corrupt at most $t$ key-issuing authorities (out of $2t + 1$) and up to $f$ (out of $3f + 1$) storage authorities in every epoch. We say that authorities corrupted by the adversary are Byzantine or faulty and the rest are honest or correct. Byzantine authorities may act arbitrarily, while correct ones follow the protocol.

**Assumption 3: Cryptography.** The cryptographic schemes used in Arke assume the existence of a non-degenerate and efficiently computable bilinear map $e : \mathbb{G}_1 \times \mathbb{G}_2 \rightarrow \mathbb{G}_T$ for which the decisional bilinear Diffie-Hellman (DBDH) assumption holds. Hash functions are modeled as random oracles and block ciphers as ideal ciphers[4]. We assume the existence of zero-knowledge non-interactive proofs (or arguments) of knowledge for NP relations.

**Assumption 4: Network model.** To capture real-world networks we assume that links between users and correct authorities are reliable (the authorities do not communicate with each other). That is, all messages among the correct authorities eventually arrive. We assume a known $\Delta$ and say that execution of a protocol is eventually synchronous if there is a global stabilization time (GST) after which all messages sent among honest parties are delivered within the network delay $\Delta$ time. An execution is synchronous if GST occurs at time 0, and asynchronous if GST never occurs. Arke assumes an partially-synchronous network. We assume that messages between users and storage authorities are anonymous. In practice, the unlinkable handshake requires that users query the storage via an anonymity network such as Tor [77] or Nym [36] (as discussed in [47]).

**Assumption 5: Roughly synchronized clocks.** Arke assumes that users have roughly synchronized clocks with the correct storage authorities. This is an assumption on the hardware run by users and authorities. We assume they can properly measure time (with some tolerance for precision) and use this ability as an out-of-band mean for synchronization. We believe this assumption is practical as many devices provide such hardware.

Definition 1 (Roughly Synchronized Clocks). *While a user is in epoch Epoch, correct authorities are either in epoch Epoch, Epoch*$-1$*, or Epoch* $+ 1$*. Also, users remain in the same epoch of each correct authority for a duration of at least* $3\Delta$ *(where* $\Delta$ *is the bound on message propagation time during periods of synchrony introduced in assumption 4).*

# 3 Preliminaries

For a security parameter $\lambda$, let $\mathbb{G}_1$, $\mathbb{G}_2$ and $\mathbb{G}_T$ be groups of prime order $q > 2^\lambda$ such that there exists an efficiently computable and non-degenerate bilinear map $e : \mathbb{G}_1 \times \mathbb{G}_2 \rightarrow \mathbb{G}_T$. We denote by $g_1, g_2$, and $g_T$ the canonical generators of $\mathbb{G}_1$, $\mathbb{G}_2$, and $\mathbb{G}_T$, respectively, and by $H : \{0,1\}^* \rightarrow \{0,1\}^l$, $H_1 : \{0,1\}^* \rightarrow \mathbb{G}_1$, and $H_2 : \{0,1\}^* \rightarrow \mathbb{G}_2$ hash functions. We treat $H$, $H_1$, and $H_2$ as random oracles.

## 3.1 Zero Knowledge Proofs

A zero-knowledge proof of knowledge (ZKPoK) is a tuple of algorithms, or protocols, that prove that an instance $x$ and witness $w$ are in a relation $\mathcal{R}$. Importantly, a ZKPoK allows the prover to prove that it *knows* the secret witness $w$; as opposed to simply proving the *existence* of the witness.

[4]Note that the random oracle model and ideal cipher model are equivalent [30].

We make use of two types of ZKPoK. The first proves knowledge of the discrete logarithm of some public value $y$ with respect to the canonical generator $g$. The second is a zk-SNARK[5] for generic NP relations. Note that although we could use the zk-SNARK to prove the discrete logarithm relation, the resulting protocol would be much more computationally expensive for the prover.

*Schnorr DLOG.* For a group $\mathbb{G}$ of prime order $q$, the Schnorr DLOG ZKPoK is a $\Sigma$-protocol for the relation

$$\mathcal{R}_{\text{DLOG}} := \left\{((x, y), \alpha) : y = x^{\alpha}\right\}$$

where $x, y \in \mathbb{G}$ and $\alpha \in \mathbb{Z}_q$. It can be compiled into a non-interactive zero-knowledge proof (NIZK) using the Fiat-Shamir transform. We denote the resulting algorithms as:

- $\text{DLOG.Prove}((x, y), \alpha) \rightarrow \pi$. Given an instance $(x, y)$ and the corresponding witness $\alpha$ such that $((x, y), \alpha) \in \mathcal{R}$, output a proof $\pi$.
- $\text{DLOG.Verify}((x, y), \pi) \rightarrow \{0, 1\}$. Given the instance $(x, y)$ and proof $\pi$, return 1 if the proof is valid and 0 otherwise.

*zk-SNARK for Hash Pre-images.* A SNARK is defined as a quadruple of algorithms $\Pi_{\mathcal{R}}$:

- $\text{Setup}(1^{\lambda}) \rightarrow (\text{crs}, \text{td})$. The Setup algorithm produces a *common reference string* crs and a *trapdoor* td.
- $\text{Prove}(\text{crs}, x, w) \rightarrow \pi$. Given the common reference string and an instance-witness pair $(x, w) \in \mathcal{R}$, output a proof $\pi$.
- $\text{Verify}(\text{crs}, x, \pi) \rightarrow \{0, 1\}$. Given the common reference string, instance, and proof, return 1 if the proof is valid and 0 otherwise.
- $\text{Simulate}(\text{crs}, \text{td}, x) \rightarrow \pi$. Using the common reference string and the trapdoor, produce a proof for the instance $x$ *without knowledge of a corresponding witness.*

The main security properties of a SNARK are *perfect completeness* and *knowledge soundness.* Perfect completeness states that a prover that knows a valid witness for the instance $x$ will always be able to produce an accepting proof. Knowledge soundness states that if a proof was accepted, then it holds with overwhelming probability that the prover knew a valid witness. A SNARK is said to be *zero-knowledge* if proofs produced by Prove and Simulate have (almost) identical probability distributions. We use the acronym *zk-SNARK* to specify that a SNARK upholds the zero-knowledge property. We use a zk-SNARK to keep users' identities private while still attesting that hashed values are correct. Let id be an identifier and $\alpha \in \mathbb{Z}_q$ a blinding factor, we define the relation $\mathcal{R}_{\text{ID}}$ as:

$$\mathcal{R}_{\text{ID}} := \left\{\left(\widehat{\text{id}}, (\text{id}, \alpha)\right) : \widehat{\text{id}} = \left(H_1(\text{id})^{\alpha}, H_2(\text{id})^{\alpha}\right)\right\}$$

For our benchmarks, we instantiate the zk-SNARK for $\mathcal{R}_{\text{ID}}$ using Groth16 [42].

## 3.2 Distributed Key Generation

A distributed key generation (DKG) protocol allows $n$ participants to jointly compute shares of a master secret without needing to compute, reconstruct or store this secret. The DKG can be parametrized with respect to a threshold $t$: any subset of at least $t + 1$ participants can perform actions that would normally require knowledge of the secret key; on the other hand, any smaller subsets cannot. A DKG can be used as a stand-in replacement for a classical key generation algorithm if it upholds the *correctness* and *secrecy* properties of Gennaro *et al.* [41].

## 3.3 Identity-Based Non-Interactive Key Exchange

*SOK ID-NIKE.* We recall the definition of the Sakai-Ohgishi-Kasahara ID-NIKE (SOK ID-NIKE) [71] in the asymmetric pairing setting, as presented in [37].

Definition 2 (Sakai-Ohgishi-Kasahara ID-NIKE [37, 71]). *The Sakai-Ohgishi-Kasahara identity-based key exchange consists of three efficiently computable algorithms* Setup, Extract, *and* SharedKey*:*

- $\text{Setup}(1^{\lambda})$*: Choose a random* $\text{msk} \xleftarrow{\$} \mathbb{Z}_q$ *and output* msk.
- $\text{Extract}(\text{msk}, \text{id})$*: compute* $d_l = H_1(\text{id})^{\text{msk}}$ *and* $d_r = H_2(\text{id})^{\text{msk}}$. *Output* $sk_{\text{id}} = (d_l, d_r)$.
- $\text{SharedKey}(sk_{\text{id}}, \text{id}')$*: We assume that identifiers are lexicographically ordered. Parse* $sk_{\text{id}}$ *as* $(d_l, d_r)$ *and output* $k_{\text{id,id}'}$*:*

$$k_{\text{id,id}'} = \begin{cases} e(d_l, H_2(\text{id}')), & \text{if } \text{id} < \text{id}' \\ e(H_1(\text{id}'), d_r), & \text{if } \text{id} > \text{id}' \end{cases}$$

*Note that* $\text{SharedKey}(sk_{\text{id}}, \text{id}') = \text{SharedKey}(sk'_{\text{id}}, \text{id})$ *for all* $\text{id} \neq \text{id}'$ *and* pp *generated by* Setup.

The security notion for such schemes is that of "indistinguishability of shared keys" [37, 61]. In the IND-SK game, an adversary is tasked with distinguishing between the shared key for a pair of identities $(\text{id}_*, \text{id}'_*)$ and a random element from the key space, in this case, $\mathbb{G}_T$. The adversary may request identity keys and shared keys from its oracles. The security game is formalized in Figure 2. We say that an ID-NIKE scheme $\Sigma$ is IND-SK secure if for any probabilistic polynomial-time adversary $\mathcal{A}$:

$$\Pr\left[\text{Exp}^{\text{IND-SK}}_{\Sigma,\mathcal{A}}(\lambda) = 1\right] \leq \frac{1}{2} + \text{negl}(\lambda)$$

The ID-NIKE of Definition 2 is IND-SK secure in the random oracle model, assuming that the decisional bilinear Diffie-Hellman (DBDH) problem is hard [37, 61].

## 3.4 Authenticated Encryption with Associated Data (AEAD)

Authenticated Encryption with Associated Data (AEAD) is a symmetric key primitive that encrypts and authenticates a message. Senders may choose to associate context data to the ciphertext in a cryptographically binding way. An AEAD scheme is defined by the following algorithms:

- $\text{AEAD.Enc}_k(m, d) \rightarrow (c, \text{tag})$. Given a key $k$, message $m$, and associated data $d$, encrypt $m$ to produce the ciphertext $c$. Authenticate the associated data and ciphertext to produce a tag tag. Output $(c, \text{tag})$.
- $\text{AEAD.Dec}_k(c, \text{tag}) \rightarrow m'$. Given a key $k$, ciphertext $c$, and associated data tag, verify the authenticity of the associated data and ciphertext. If the verification rejects, output $m' \leftarrow \perp$. Otherwise decrypt $c$ and output $m' \leftarrow m$.

[5] Zero-knowledge succinct non-interactive argument of knowledge

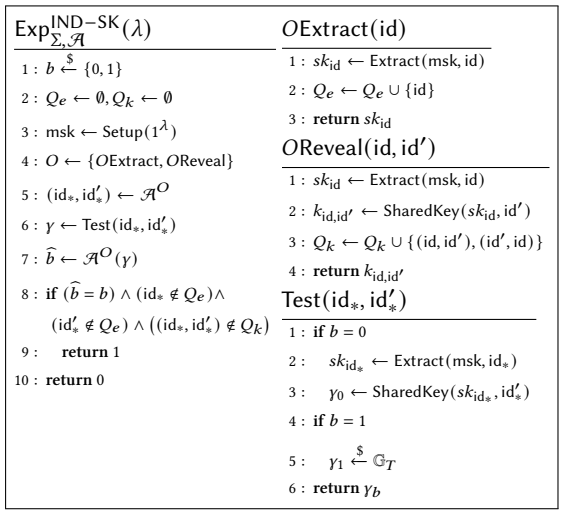


**Figure 2: IND-SK security game for ID-NIKEs**

## 4 The Arke Contact Discovery Protocol

The Arke contact discovery protocol combines an ID-NIKE scheme with an unlinkable handshake. The ID-NIKE allows users to establish shared secrets amongst each other knowing only their (potentially low-entropy) identifiers. Using this shared secret, they can run the unlinkable handshake to exchange arbitrary messages through an untrusted key-value store. We describe a private and trust-minimized variant of the SOK ID-NIKE (Section 4.1), and an unlinkable handshake protocol (Section 4.2), and show how to combine both to build a contact discovery protocol (Section 4.3).

### 4.1 Threshold Oblivious ID-NIKE

The ID-NIKE of Sakai, Ohgishi and Kasahara [71] relies on a trusted third party to issue private keys to users. We modify their protocol to meet our privacy desiderata by *(i)* allowing users to verify the private keys they are issued, *(ii)* separating the key issuance operation into a registration and an extraction phase and, *(iii)* distributing the master secret key. We achieve modifications *(i)* and *(iii)* by applying techniques outlined by Boneh and Franklin [16]; we achieve modification *(ii)* by improving upon the result of Sui *et al.* [73]. We refer to the resulting protocol as a threshold and oblivious ID-NIKE.

*Verifiable key issuance.* One way to hold the trusted third party accountable is to allow other parties in the system to verify the issuance of private keys. To this effect, we modify the Setup algorithm to output a master public key mpk and introduce the VerifyPK and VerifyExtract algorithms:

- $\mathsf{Setup}(1^\lambda) \rightarrow (\mathsf{msk}, \mathsf{mpk})$: choose a random $\mathsf{msk} \xleftarrow{\$} \mathbb{Z}_q$ and compute the corresponding public key $\mathsf{mpk} = (g_1^{\mathsf{msk}}, g_2^{\mathsf{msk}})$. Output msk and mpk.
- $\mathsf{VerifyPK}(\mathsf{pk}) \rightarrow \{0,1\}$: parse pk as $(\mathsf{pk}_l, \mathsf{pk}_r)$. If $e\left(\mathsf{pk}_l, g_2\right) = e\left(g_1, \mathsf{pk}_r\right)$, output 1 (accept). Otherwise output 0 (reject).
- $\mathsf{VerifyExtract}(\mathsf{mpk}, \mathsf{id}, \theta) \rightarrow \{0,1\}$: parse mpk as $(\mathsf{mpk}_l, \mathsf{mpk}_r)$ and $\theta$ as $(\theta_l, \theta_r) \in \mathbb{G}_1 \times \mathbb{G}_2$. If $e\left(\theta_l, g_2\right) = e\left(H_1(\mathsf{id}), \mathsf{mpk}_r\right)$ and $e\left(g_1, \theta_r\right) = e\left(\mathsf{mpk}_l, H_2(\mathsf{id})\right)$, output 1 (accept). Otherwise, output 0 (reject).

The VerifyPK algorithm enforces that the terms in the pk tuple are equal to the generators $g_1$ and $g_2$ taken to the *same* power. The VerifyExtract is analogous to the verification of a BLS signature [17]: it returns 1 if and only if the input $\theta$ is equal to the expected private key. We show that both of these algorithms behave as expected in the extended version of the paper[6].

*Oblivious key issuance.* In the SOK ID-NIKE, users must reveal their identifier to a trusted third party to obtain their secret key. We follow the approach of Sui *et al.* [73] and separate this trusted party into two entities: a registration authority and a key-issuing authority. We allow the registration authority to learn identifiers but not to compute their private keys. Its role is to attest that a user $A$ owns the identifier $\mathsf{id}_A$. The key-issuing authority is however able to produce private keys but does not learn which identities have requested keys.

To this effect, we introduce a setup algorithm for the registration authority, $\mathsf{Setup}_R$, and replace the Extract algorithm by five efficiently computable algorithms Register, Blind, VerifyID, BlindExtract and Unblind:

- $\mathsf{Setup}_R$: Produces private and public parameters for a registration authority.
- Register: Upon valid authentication, a registration authority produces a signature attesting that user $A$ owns the identifier $\mathsf{id}_A$.
- Blind: Produce a masked version of an identifier and its corresponding registration signature. The blinded identifier and signature are accompanied by optional proof of their validity.
- VerifyID: Verify that a valid registration signature was issued for a blinded identifier.
- BlindExtract: Given a blinded identifier, produce the corresponding blinded secret key.
- Unblind: Recover an identifier's secret key from a blinded secret key.

We give a concrete construction of an oblivious ID-NIKE in the extended version of the paper. Our construction can be seen as an improvement over that of Sui *et al.* [73].

*Distributed key issuance.* In the oblivious setting described above, the key-issuing authority is still all-powerful in that it is able to extract the private key of any identifier. To minimize the trust placed in the key-issuing authority, we distribute it into $n$ entities that each hold a share of the master secret key. Using a $(t, n)$-threshold DKG, we ensure that the ID-NIKE remains IND-SK secure when no more than $t$ parties are malicious.

We distribute the key-issuing authority by replacing the $\mathsf{Setup}_E$ algorithm with a secure DKG [41]. The extraction algorithm is the same as BlindExtract but is renamed to BlindPartialExtract to emphasize the fact that it outputs blinded *partial* secret keys. Similarly, the verification of a partial private key is identical to VerifyExtract but is renamed to VerifyPartialExtract. Finally, we introduce the Combine algorithm to reconstruct a secret key from a set of $t + 1$ key shares.

[6]The extended version of the paper is available at https://eprint.iacr.org/2023/1218

Definition 3 (Threshold and Oblivious ID-NIKE). *Let $\Pi_{\mathsf{ID}}$ be a knowledge sound zk-SNARK for the relation $\mathcal{R}_{\mathsf{ID}}$. We define the $(t,n)$-threshold variant of the oblivious SOK ID-NIKE as follows:*

- **$\mathsf{SetupDKG}_E(1^\lambda, t, n) \rightarrow (\mathsf{msk}_1, \ldots, \mathsf{msk}_n, \mathsf{pp})$**. *All n participants $P_1, \ldots, P_n$ jointly execute a secure DKG to compute Shamir secret shares $\mathsf{msk}_1, \ldots, \mathsf{msk}_n$ of an (unknown) master secret key* msk. *They jointly output a* transcript, *a set of partial public keys $\{\mathsf{mpk}_i = (g_1^{\mathsf{msk}_i}, g_2^{\mathsf{msk}_i})\}_{i=1}^n$ and master public key* $\mathsf{mpk} = (g_1^{\mathsf{msk}}, g_2^{\mathsf{msk}})$. *Output* $\mathsf{msk}_i$ *to* $P_i$ *and* $\mathsf{pp} \leftarrow (\mathsf{transcript}, \mathsf{mpk})$.
- **$\mathsf{Setup}_R(1^\lambda, \mathsf{pp}) \rightarrow (\mathsf{rsk}, \mathsf{pp})$**. *Choose a random registration secret key* $\mathsf{rsk} \xleftarrow{\$} \mathbb{Z}_q$ *and compute the registration public key* $\mathsf{rpk} = (g_1^{\mathsf{rsk}}, g_2^{\mathsf{rsk}})$. *Output* rsk *and* $\mathsf{pp} \leftarrow \mathsf{pp}||\mathsf{rpk}$.
- **$\mathsf{VerifyPK}(\mathsf{pk}) \rightarrow \{0,1\}$**. *Parse* pk *as* $(\mathsf{pk}_l, \mathsf{pk}_r)$. *If* $e(\mathsf{pk}_l, g_2) = e(g_1, \mathsf{pk}_r)$, *output* 1 *(accept). Otherwise output* 0 *(reject).*
- **$\mathsf{Register}(\mathsf{rsk}, \mathsf{id}) \rightarrow \tau_{\mathsf{id}}$**. *Compute* $\tau_l = H_1(\mathsf{id})^{\mathsf{rsk}}$ *and* $\tau_r = H_2(\mathsf{id})^{\mathsf{rsk}}$. *Output the registration signature* $\tau_{\mathsf{id}} = (\tau_l, \tau_r)$.
- **$\mathsf{Blind}(\mathsf{pp}, \mathsf{id}, \tau_{\mathsf{id}}) \rightarrow (\alpha, \widehat{\mathsf{id}}, \widehat{\tau_{\mathsf{id}}}, \pi)$**. *Sample a random blinding factor* $\alpha \xleftarrow{\$} \mathbb{Z}_q$. *Compute*

$$\begin{aligned} \widehat{\mathsf{id}} &= \left(H_1(\mathsf{id})^\alpha, H_2(\mathsf{id})^\alpha\right) \\ \pi &= \Pi_{\mathsf{ID}}.\mathsf{Prove}(\mathsf{pp}_{\mathsf{ZK}}, \widehat{\mathsf{id}}, (\mathsf{id}, \alpha)) \\ \widehat{\tau_{\mathsf{id}}} &= \tau_{\mathsf{id}}{}^\alpha \end{aligned} \tag{1}$$

  *Output the blinding factor $\alpha$, blind identifier* $\widehat{\mathsf{id}}$, *blind registration signature* $\widehat{\tau_{\mathsf{id}}}$ *and the blinding proof* $\pi$.
- **$\mathsf{VerifyID}(\mathsf{pp}, \widehat{\mathsf{id}}, \widehat{\tau_{\mathsf{id}}}, \pi) \rightarrow \{0,1\}$**. *Parse* rpk *as* $(pk_l, pk_r)$, $\widehat{\mathsf{id}}$ *as* $(\widehat{\mathsf{id}}_l, \widehat{\mathsf{id}}_r)$, *and* $\widehat{\tau_{\mathsf{id}}}$ *as* $(\widehat{\tau_l}, \widehat{\tau_r})$. *Check that the following equations hold:*

$$\begin{aligned} e(\widehat{\tau_l}, g_2) &\stackrel{?}{=} e\left(\widehat{\mathsf{id}}_l, pk_r\right) \\ e(g_1, \widehat{\tau_r}) &\stackrel{?}{=} e\left(pk_l, \widehat{\mathsf{id}}_r\right) \\ \Pi_{\mathsf{ID}}.\mathsf{Verify}(\mathsf{pp}_{\mathsf{ZK}}, \mathsf{ID}, \pi_{\mathsf{ID}}) &\stackrel{?}{=} 1 \quad \textit{(accept)} \end{aligned} \tag{2}$$

  *If all equations verify output* 1, *otherwise output* 0.
- **$\mathsf{BlindPartialExtract}(\mathsf{msk}_i, \widehat{\mathsf{id}}) \rightarrow \widehat{sk_{\mathsf{id},i}}$**. *Compute and output the blind secret key share* $\widehat{sk_{\mathsf{id},i}} = \widehat{\mathsf{id}}^{\mathsf{msk}_i}$.
- **$\mathsf{Unblind}(\widehat{sk_{\mathsf{id},i}}, \alpha) \rightarrow sk_{\mathsf{id},i}$**. *Compute and output the partial key* $sk_{\mathsf{id},i} = \widehat{sk_{\mathsf{id},i}}^{\frac{1}{\alpha}}$.
- **$\mathsf{VerifyPartialExtract}(\mathsf{mpk}_i, \mathsf{id}, \theta)$**. *Parse* $\mathsf{mpk}_i$ *as* $(\mathsf{mpk}_{i,l}, \mathsf{mpk}_{i,r}) \in \mathbb{G}_1 \times \mathbb{G}_2$ *and* $\theta$ *as* $(\theta_l, \theta_r) \in \mathbb{G}_1 \times \mathbb{G}_2$. *If* $e(\theta_l, g_2) = e\left(H_1(\mathsf{id}), \mathsf{mpk}_{i,r}\right)$ *and* $e(g_1, \theta_r) = e\left(\mathsf{mpk}_{i,l}, H_2(\mathsf{id})\right)$, *output* 1 *(accept). Otherwise, output* 0 *(reject).*
- **$\mathsf{Combine}(\{sk_{\mathsf{id},i}\}_{i=1}^{t+1}) \rightarrow sk_{\mathsf{id}}$**. *Using a set of $t+1$ valid partial keys, compute $d_l$ and $d_r$ using Lagrange interpolation "in the exponent". Let $L_i$ denote the Lagrange coefficient for the i-th share in the given set, $d_l = \prod_{i=1}^{t+1} {d_{l,i}}^{L_i}$ and $d_r = \prod_{i=1}^{t+1} {d_{r,i}}^{L_i}$.* [7] *Output the user key* $sk_{\mathsf{id}} = (d_l, d_r)$.
- **$\mathsf{SharedKey}(sk_{\mathsf{id}}, \mathsf{id}') \rightarrow k_{\mathsf{id},\mathsf{id}'}$**. *We assume that identifiers are lexicographically ordered. Parse* $sk_{\mathsf{id}}$ *as* $(d_l, d_r)$ *and output* $k_{\mathsf{id},\mathsf{id}'}$:

$$k_{\mathsf{id},\mathsf{id}'} = \begin{cases} e(d_l, H_2(\mathsf{id}')), & \textit{if } \mathsf{id} < \mathsf{id}' \\ e(H_1(\mathsf{id}'), d_r), & \textit{if } \mathsf{id} > \mathsf{id}' \end{cases}$$

*For all* $\mathsf{id} \neq \mathsf{id}'$ *and* pp *generated by* $\mathsf{SetupDKG}_E$, *it holds that* $\mathsf{SharedKey}(\mathsf{pp}, sk_{\mathsf{id}}, \mathsf{id}') = \mathsf{SharedKey}(\mathsf{pp}, sk'_{\mathsf{id}}, \mathsf{id})$ .

*IND-SK security.* We show that the threshold and oblivious ID-NIKE of Definition 3 is IND-SK secure under the DBDH assumption in the random oracle model if $\Pi_{\mathsf{ID}}$ is a knowledge sound SNARK for $\mathcal{R}_{\mathsf{ID}}$.

Theorem 1. *The threshold and oblivious ID-NIKE of Definition 3 is IND-SK under the DBDH assumption when modeling the functions $H_1, H_2$ as random oracles, and if $\Pi_{\mathsf{ID}}$ is a knowledge sound SNARK for $\mathcal{R}_{\mathsf{ID}}$.*

*Proof intuition.* We provide intuition for the proof of Theorem 1; a full proof is presented in the extended version of the paper[8]. The proof first shows that the (centralized) oblivious variant of the SOK ID-NIKE is IND-SK secure under the same assumptions as the classic SOK ID-NIKE if $\Pi_{\mathsf{ID}}$ is a knowledge sound SNARK for $\mathcal{R}_{\mathsf{ID}}$. We can then apply the arguments of Boneh and Franklin [16] to replace the key generation algorithm with a secure DKG.

We prove the former by showing a reduction from the classic IND-SK security game to the oblivious IND-SK game. In a nutshell, the adversary performing the reduction takes on the role of the registration authority. It samples a registration key and can naturally answer the inner adversary's Register queries. To answer BlindExtract oracle queries, the reduction must first "unblind" the queried identifier. This is done by running the extractor for $\Pi_{\mathsf{ID}}$. We show that this reduction strategy has an overwhelming success probability if $\Pi_{\mathsf{ID}}$ is a knowledge sound SNARK for $\mathcal{R}_{\mathsf{ID}}$.

*Anonymity from the key-issuing authorities.* Identifiers are kept hidden from the key-issuing authorities if $\Pi_{\mathsf{ID}}$ is a zero-knowledge SNARK for $\mathcal{R}_{\mathsf{ID}}$. We prove this claim by showing the existence of an algorithm SimulateID that *does not know an identifier* yet produces tuples $(\widehat{\mathsf{id}}_{\mathsf{sim}}, \widehat{\tau}_{\mathsf{sim}}, \pi_{\mathsf{sim}})$ which are statistically indistinguishable from tuples $(\widehat{\mathsf{id}}, \widehat{\tau_{\mathsf{id}}}, \pi)$ produced by an honest prover running Blind [27].

- **$\mathsf{SimulateID}(\mathsf{crs}, \mathsf{td}) \rightarrow (\widehat{\mathsf{id}}_{\mathsf{sim}}, \widehat{\tau}_{\mathsf{sim}}, \pi_{\mathsf{sim}})$**. Sample $\widehat{\tau}_{\mathsf{sim}} \xleftarrow{\$} \mathbb{G}_1 \times \mathbb{G}_2$ and compute:

$$\begin{aligned} \widehat{\mathsf{id}}_{\mathsf{sim}} &= \widehat{\tau}_{\mathsf{sim}} \circ \mathsf{rpk}^{-1} \\ \pi_{\mathsf{sim}} &= \Pi_{\mathsf{ID}}.\mathsf{Simulate}(\mathsf{crs}, \mathsf{td}, \widehat{\mathsf{id}}_{\mathsf{sim}}) \end{aligned}$$

By construction, the tuple $(\widehat{\mathsf{id}}_{\mathsf{sim}}, \widehat{\tau}_{\mathsf{sim}}, \pi_{\mathsf{sim}})$ satisfies the checks of VerifyID. Furthermore, since the blinding factors are sampled uniformly from $\mathbb{Z}_q$, then $(\widehat{\mathsf{id}}_{\mathsf{sim}}, \widehat{\tau}_{\mathsf{sim}})$ follow the same probability distribution as $(\widehat{\mathsf{id}}, \widehat{\tau_{\mathsf{id}}})$. Finally, by the zero-knowledge property of $\Pi_{\mathsf{ID}}$, it holds that $\pi_{\mathsf{sim}}$ is statistically indistinguishable from $\pi$.

[7] As required, $d_l = \prod_{i=1}^{t+1} {d_{l,i}}^{L_i} = H_1(\mathsf{id})^{\sum_{i=1}^{t+1} \mathsf{msk}_E, i L_i} = H_1(\mathsf{id})^{\mathsf{msk}_E}$ and analogously for $d_r$.

[8] The extended version of the paper is available at https://eprint.iacr.org/2023/1218

*Alternative construction.* In the extended version of the paper, we describe an alternative ID-NIKE construction which uses Pedersen's DKG [62] to setup the threshold committee. While the Pedersen-DKG is simple and efficient, it does not uphold the *correctness* and *secrecy* properties of Gennaro *et al.* [41] and may not always be used as a stand-in replacement for a generic key generation procedure. Instead, we prove the security of the resulting threshold ID-NIKE using the notion of key-expressable DKGs from Gurkan *et al.* [43].

## 4.2 Unlinkable Handshake

Performing an identity-based key exchange only addresses half of the contact discovery problem. Users must also exchange an initial message (or flag) in a privacy-preserving way without prior knowledge of each other's network addresses. We present an unlinkable handshake protocol over a public, untrusted message board [47]. We use the message board as a key-value store. In this section, we treat the store as a black box; Section 5 shows how to efficiently instantiate such storage with minimal trust assumptions and no single point of failure.

*Overview.* Using their shared ID-NIKE key, Alice and Bob each locally derive a "write tag", a "read tag" and an AEAD encryption key. They use the AEAD encryption key to encrypt their messages and post the resulting ciphertexts on the message board at a unique location derived from their "write tag". We allow all users and network observers to read from the store. However, only users that know read tags destined for them and the corresponding encryption key will be able to recover messages.

Definition 4. *Let $\mathbb{G}$ be an abelian group of prime order $p$ with canonical generator $g$. Let* DLOG *be a non-interactive instantiation of the Schnorr proof of discrete logarithm compiled using the Fiat-Shamir heuristic. We use a variant of the proof where an extra "context" nonce is added to the transcript. This nonce will be used to bind a proof to a specific session between the message board and a user, thus preventing replay attacks. We denote $\pi_x^{(r)}$ as a proof of knowledge of the secret exponent $x$ during session $r$.*

*Let* AEAD *be an IND-CCA secure authenticated encryption with associated data scheme. We denote $\mathcal{K}$ the set of accepted keys for this scheme and $\mathcal{C}$ the set of ciphertexts.*

*The handshake is parametrized by two functions, a* key derivation function $\mathsf{KDF} : \{0,1\}^* \to \mathcal{K}$ *and a* tag derivation function $\mathsf{TDF} : \{0,1\}^* \times \{0,1\} \to \mathbb{Z}_p$. *Assuming that every pair of users $A$ and $B$ have derived a shared secret $s_{AB}$, the unlinkable handshake is defined as:*

- $\mathbf{Write}^{(r)}(s_{AB}, \mathbf{id}_A, \mathbf{id}_B, m) \to (\mathbf{loc}_w, \pi_w^{(r)}, c)$. *Compute a symmetric key $k = \mathsf{KDF}(s_{AB})$ and tag $t_w$ such that:*

$$t_w = \begin{cases} \mathsf{TDF}(s_{AB}, 0), & \text{if } \mathsf{id}_A < \mathsf{id}_B \\ \mathsf{TDF}(s_{AB}, 1), & \text{if } \mathsf{id}_A > \mathsf{id}_B \end{cases}$$

  *Compute $\mathsf{loc}_w = g^{t_w}$. Using the derived key and tag, compute the ciphertext $c = \mathsf{AEAD.Enc}_k(g^{t_w}, m)$. Finally, for the current session $r$, compute the proof $\pi_{t_w}^{(r)} = \mathsf{DLOG.Prove}((g, loc_w), t_w, r)$. Output $(\mathsf{loc}_w, \pi_w^{(r)}, c)$.*
- $\mathbf{VerifyWrite}^{(r)}(\mathbf{loc}_w, \pi_w^{(r)}) \to \{\mathbf{0}, \mathbf{1}\}$. *Compute and output $b = \mathsf{DLOG.Verify}(\mathsf{loc}_w, \pi_w^{(r)}, r)$.*
- $\mathbf{Read}(s_{AB}, \mathbf{id}_A, \mathbf{id}_B) \to \boldsymbol{m}$. *Compute a symmetric key $k = \mathsf{KDF}(s_{AB})$ and tag $t_r$ such that:*

$$t_r = \begin{cases} \mathsf{TDF}(s_{AB}, 1), & \text{if } \mathsf{id}_A < \mathsf{id}_B \\ \mathsf{TDF}(s_{AB}, 0), & \text{if } \mathsf{id}_A > \mathsf{id}_B \end{cases}$$

  *Compute $\mathsf{loc}_r = g^{t_r}$. Retrieve the value $c'$ associated with location $\mathsf{loc}_r$ in the store. Compute $m = \mathsf{AEAD.Dec}_k(c', \mathsf{loc}_r)$.*

*Importantly, $A$ and $B$ can derive the same AEAD symmetric key. Furthermore, $A$'s read tag matches the definition of $B$'s write tag (and conversely).*

*The handshake is said to be complete when a pair of users have both performed the* Write *and* Read *operations. Let $t_A, c_A$ and $t_B, c_B$ be the write tags and ciphertexts derived by $A$ and $B$ respectively, we define the transcript of a completed handshake as:*

$$\mathsf{tr} \leftarrow (r, r', g^{t_A}, g^{t_B}, \pi_{t_A}^{(r)}, \pi_{t_B}^{(r')}, c_A, c_B)$$

*Confidentiality.* The handshake described above can be shown to preserve the confidentiality of the underlying messages. Indeed if KDF is a secure pseudorandom function, then the derived symmetric key $k_{AB}$ is indistinguishable from random. This in turn allows us to uphold the IND-CCA property of the AEAD scheme.

*Unlinkability.* To meet our privacy goals, we need to ensure that observing a transcript does not leak information about the identities of the users that generated it. This property should still hold even if the adversary controls all other identities identities and is successful in completing handshakes with each of the target users. Furthermore, we assume that each identity has a *fixed* message that it tries to communicate.

We capture this security notion by defining an *unlinkability* game (see Figure 3). An adversary $\mathcal{A}$ is tasked with distinguishing between a transcript for the pair of identities $\mathsf{id}_*, \mathsf{id}'_*$ and a random transcript. The adversary is allowed to query any shared secret or valid transcripts, and may even complete valid handshakes with both of the target identities.

We say that a handshake HS is unlinkable if for any probabilistic polynomial-time adversary $\mathcal{A}$:

$$\Pr\left[\mathsf{Exp}^{\mathsf{Unlinkability}}_{\mathsf{HS},\mathcal{A}}(\lambda) = 1\right] \leq \frac{1}{2} + \mathsf{negl}(\lambda)$$

Theorem 2. *The handshake presented in Definition 4 is unlinkable if shared secrets between users are established using an IND-SK secure ID-NIKE.*

Proof (Theorem 2). Let $\Sigma$ denote a secure ID-NIKE. Assume for the sake of contradiction that there exists an adversary $\mathcal{A}$ for which $\Pr\left[\mathsf{Exp}^{\mathsf{Unlinkability}}_{\mathsf{HS},\mathcal{A}}(\lambda) = 1\right] > \frac{1}{2} + \mathsf{negl}(\lambda)$.

We construct an adversary $\mathcal{B}$ that runs $\mathcal{A}$ as a sub-routine against the IND-SK game (Figure 2). Let $T_M$ be a table mapping identifiers to messages. $T_M$ is initialized as the empty table. $\mathcal{B}$ simulates any call to the function $M$ (line 3 of $O$Transcript and line 6 of Test) by running the following SimMessage routine: if $\mathsf{id} \in T_M$, return $T_M[\mathsf{id}]$; else, $m \xleftarrow{\$} \mathcal{M}$, write $T_M[\mathsf{id}] \leftarrow m$ and return $m$. $\mathcal{B}$ simulates $\mathcal{A}$'s oracles as follows:

- $O$Secret: replace line 1 of the $O$Secret procedure by a call to $O$Reveal.

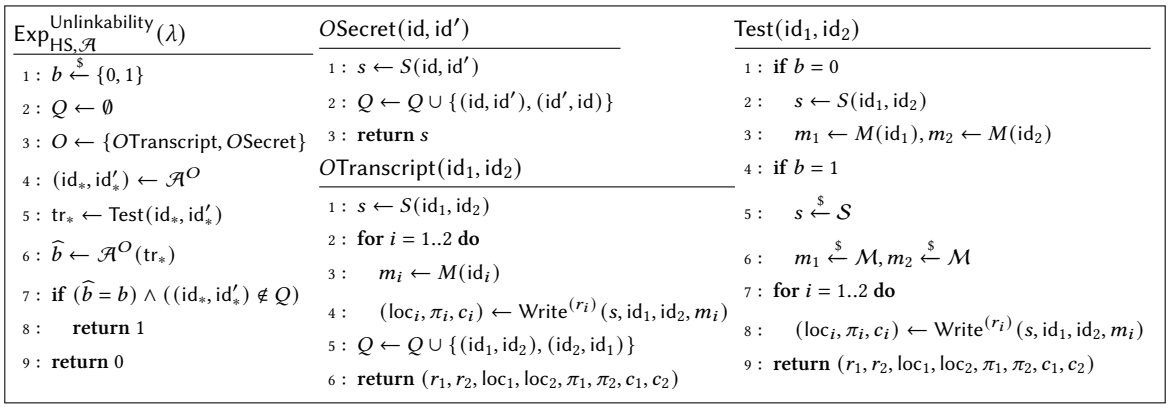


**Figure 3: Unlinkability game. Here $\mathcal{M}$ and $\mathcal{S}$ respectively denote the set of messages and shared secrets. Similarly, $M : \mathcal{I} \to \mathcal{M}$ and $S : \mathcal{I} \times \mathcal{I} \to \mathcal{S}$ denote the implicit maps from identities to messages and shared secrets. We assume that $S(a, b) = S(b, a)$.**

- $O$Transcript: replace line 1 of the $O$Transcript procedure by a call to $O$Reveal. Replace line 3 with a call to SimMessage.
- Test: $\mathcal{B}$ returns the same identity pair $\mathsf{id}_*, \mathsf{id}'_*$ that $\mathcal{A}$ outputs (line 4 of the game's code) and receives the value $\gamma$. Call SimMessage for each of the provided identities. Perform the loop of lines 7 and 8 of the test procedure replacing $s$ by $\gamma$.

Notice that after all of $\mathcal{A}$'s queries, it holds that the exclusion sets of both games are equal. Indeed every update to $Q$ generated the same update to $Q_k$ and no queries were made to $\mathcal{B}$'s $O$Extract oracle. Therefore, $Q_e = \emptyset$. Furthermore, by definition of $\mathsf{Exp}^{\mathsf{IND-SK}}_{\Sigma,\mathcal{B}}(\lambda)$, $s$ and $\gamma$ follow the same distribution. Therefore:

$$\Pr\left[\mathsf{Exp}^{\mathsf{IND-SK}}_{\Sigma,\mathcal{B}}(\lambda) = 1\right] = \Pr\left[\mathsf{Exp}^{\mathsf{Unlinkability}}_{\mathsf{HS},\mathcal{A}}(\lambda) = 1\right]$$

We have shown that $\mathcal{B}$ gains a non-negligible advantage in the IND-SK game against the secure ID-NIKE $\Sigma$, therefore reaching a contradiction. Thus, for a secure ID-NIKE scheme $\Sigma$ there exists no PPT adversary $\mathcal{A}$ such that $\Pr\left[\mathsf{Exp}^{\mathsf{Unlinkability}}_{\mathsf{HS},\mathcal{A}}(\lambda) = 1\right] > \frac{1}{2} + \mathsf{negl}(\lambda)$. Therefore, HS is an unlinkable handshake. □

Note that the unlinkability game does not consider network adversaries. In fact, read/write patterns to the bulletin board do leak information about relations between users [47]. To prevent such attacks, we require that each read to- and each write from the bulletin board be performed through an anonymity network, using a fresh identity (see Assumption 4).

*Bilateral handshake.* An important property of our handshake is that it is *bilateral*: each user may choose to participate or withhold from performing the handshake with a given user. In that sense, the adversary in the unlinkability game is stronger than most real-world adversaries. Indeed in the unlinkability game, the adversary may coerce any user into performing the handshake with her. In practice, the bilateral property of the handshake protects our system from "crawling attacks" as studied by Hagen *et al.* [44].

*Overwrite protection.* If TDF is a collision-resistant hash function (CRHF) then write and read tags may only be derived by users that know the relevant shared seed (except for a very unlikely collision). This in turn implies that only users that know a shared seed are able to produce a valid ZKPoK for the relevant tag. Thus verifying the ZKPoK in the Write protocol enforces *access control* for a given write location.

*Bounded storage.* Unfortunately, this access control is not enough to prevent a malicious user from filling up the message board with fake messages. This adversary can pick random tag values and produce valid proofs for those. The mitigation strategy depends on the nature of the store authorities. Protecting our custom-built store authorities (Section 5) against those attacks requires the introduction of a privacy-preserving rate-limiting mechanism such as PrivacyPass [34] or the construction of Camenisch *et al.* [21]. If the store authorities coincide with the maintainers of an existing blockchain (Section 5.2), the native token required to pay for the blockchains' gas cost effectively acts as a rate-limiting mechanism. As a result, Arke does not need to introduce any new access control mechanism.

## 4.3 Contact Discovery

Let $\mathcal{R}_{\mathsf{domID}}$ be a variant of $\mathcal{R}_{\mathsf{ID}}$ where part of the hash functions' input is public:

$$\mathcal{R}_{\mathsf{domID}} := \left\{ \begin{array}{l} \left((\widehat{\mathsf{id}}, \mathsf{dom}), (\mathsf{id}, \alpha)\right) : \\ \widehat{\mathsf{id}} = (H_1(\mathsf{id}||\mathsf{dom})^\alpha, H_2(\mathsf{id}||\mathsf{dom})^\alpha) \end{array} \right\}$$

Let ID-NIKE designate the threshold and oblivious ID-NIKE of Definition 3 where $\Pi_{\mathsf{ID}}$ is replaced with a proof $\Pi_{\mathsf{domID}}$ for $\mathcal{R}_{\mathsf{domID}}$, and HS designate the unlinkable handshake of Definition 4.

We define the contact discovery protocol for a registration authority RA, key-issuing committee $(\mathsf{KA}_1, \ldots, \mathsf{KA}_n)$, user $\mathcal{U}$ and bulletin board BB as follows:

(1) $\mathcal{U} \leftrightarrow$ RA: $\mathcal{U}$ and RA engage in an authentication protocol (defined by RA) to prove that the identifier $\mathsf{id}_\mathcal{U}$ belongs to $\mathcal{U}$. Upon successful completion, RA sends $\tau_\mathcal{U} = \text{ID-NIKE.Register}(\mathsf{rsk}_{\mathsf{RA}}, \mathsf{id}_\mathcal{U}||\mathsf{dom})$.
(2) $\mathcal{U} \leftrightarrow \mathsf{KA}_i$, for up to $2t + 1$ key-issuing authorities (and a minimum of $t + 1$ in the ideal case): $\mathcal{U}$ computes

$$(\alpha, \widehat{\mathsf{id}_\mathcal{U}}, \widehat{\tau_\mathcal{U}}, \pi) = \text{ID-NIKE.Blind}(\mathsf{pp}, (\mathsf{id}_\mathcal{U}||\mathsf{dom}), \tau_\mathcal{U})$$

and sends the blind key-issuance request $(\widehat{\mathsf{id}_{\mathcal{U}}}, \widehat{\tau_{\mathcal{U}}}, \pi)$. If ID-NIKE.VerifyID(pp, $(\widehat{\mathsf{id}_{\mathcal{U}}}, \mathsf{dom}), \widehat{\tau_{\mathcal{U}}}, \pi) = 1$, $\mathsf{KA}_i$ sends the blind secret key share ID-NIKE.BlindPartialExtract($\mathsf{msk}_i, \widehat{\mathsf{id}_{\mathcal{U}}}$).

(3) $\mathcal{U}$, one-time local operation: let $\widehat{sk_i}$ and $\alpha_i$ denote the $i$-th blind share and the $i$-th blinding factor, $\mathcal{U}$ computes:

$$sk_i = \text{ID-NIKE.Unblind}(\widehat{sk_i}, \alpha_i)$$
$$sk = \text{ID-NIKE.Combine}(\{sk_i\}_{i=1}^{t+1})$$

(4) $\mathcal{U}$, locally, for each contact identifier $\mathsf{id}_C$: compute a share secret $s_{\mathcal{U},C}$ As

$$s_{\mathcal{U},C} = \text{ID-NIKE.SharedKey}(sk, \mathsf{id}_C)$$

(5) $\mathcal{U} \leftrightarrow$ BB, store write for each contact $\mathsf{id}_C$: $\mathcal{U}$ sends a write request

$$(\mathsf{loc}_w, \pi_w^{(r)}, c) = \mathsf{HS.Write}^{(r)}(s_{\mathcal{U},C}, \mathsf{id}_{\mathcal{U}}, \mathsf{id}_C, m)$$

If $\mathsf{HS.VerifyWrite}^{(r)}(\mathsf{loc}_w, \pi_w^{(r)}) = 1$, BB writes $c$ in the location $\mathsf{loc}_w$.

(6) $\mathcal{U} \leftrightarrow$ BB, store read for each contact $\mathsf{id}_C$: $\mathcal{U}$ and BB perform HS.Read.

For clarity, this definition omits checking the correctness of the key shares (performed by the user), that the public key of the registration authority maps to its recognized domain (performed by the store authorities), and the validity of the rate-limiting tokens (performed by the store authorities, see Section 4.2).

*Discovery epochs.* Taking advantage of the roughly synchronized clocks (see Assumption 5), we can define discovery epochs of fixed duration (e.g., one week or one month). At the end of each epoch, store entries can be wiped. This allows the store to drop any values that are left behind after a complete handshake. On the other hand, handshakes that were only partially completed during such an epoch are aborted and will require users to once again perform the discovery phase.

*Forward secrecy.* Although we have shown that messages on the store are securely encrypted, the Arke protocol does not provide confidentiality if the system is compromised. Indeed, the AEAD symmetric key is deterministically computed from the shared secret derived using an ID-NIKE. As shown by Paterson and Srinivasan [61], ID-NIKEs do not provide forward secrecy. To mitigate such risks, we recommend that users only include "public" information (public keys, wallet address, etc.) in their initial message, and use it to establish an out-of-bound communication channel.

## 5 The Arke Key-Value Store

We present two types of distributed stores that fulfill the required properties set in Section 2.3. Section 5.1 presents a custom store designed to be run by large messaging companies such as WhatsApp, Signal, and Telegram across multiple data centers. Section 5.2 illustrates how to leverage existing (production-ready) blockchains as Arke store without requiring any modification to their protocol.

### 5.1 Custom Arke Store

This store provides extremely low latency by forgoing consensus and instead leveraging simpler and more efficient broadcast-based primitives (based on Consistent Broadcast [20]). This store is designed to sustain a Byzantine adversary (to withstand partially corrupt store operators) but the extended version of the paper[9] shows a straightforward conversion into a crash fault-tolerant store. The extended version additionally details the protocol messages and data structures run by the store's nodes, provides complete algorithms, explains how to clean up storage, and how to scale the system by maximizing parallel processing of transactions and leveraging more hardware to increase its capacity. We also formally prove the validity, consistency, and termination of this store protocol.

Figure 4 presents an overview of the protocol allowing user $A$ to respectively write and read the key-value pairs $(\mathsf{loc}_{AB}, c_{AB})$ and $(\mathsf{loc}_{BA}, c_{BA})$ from the store.

*Writing the store.* Steps ❶-❸ of Figure 4 illustrate the high-level interactions between user $A$ and the storage authorities to allow the user to write the distributed store. User $A$ uses its writing tag $t_{AB}$ (Section 4.2) as a private signing key to create and sign a *write transaction*. This transaction mutates (or creates) the key-value pair $(\mathsf{loc}_{BA}, c_{BA}) = (g_1^{t_{BA}}, c_{BA})$ of the Arke store (❶). The user transaction is then sent to each Arke storage authority (❷). The authorities check it for validity and lock the store entry to mutate (❸). The write operation is completed as soon as $2f + 1$ authorities successfully terminate this step. A detailed description of how authorities process incoming write transactions is included in the appendix of the extended paper.

*Synchronization.* Steps ❹-❼ of Figure 4 illustrate the store synchronization step. At this stage, user signature keys are not needed anymore, and the synchronization process may be performed by any user client or third-party synchronizer process. Storage authorities always provide idempotent replies to protocol messages: it is safe to send multiple times the same message to an authority. After processing a write transaction, each authority returns a *vote* to the user or synchronizer process (❹). The user collects the votes from a quorum of $2f+1$ authorities to form a *certificate* (❺). The certificate is then sent back to all validators (❻). The authorities check the certificate and upon success mutate the specified store entry and release the locks to allow future updates (❼). A detailed description of this step is included in the appendix of the extended paper. The write and synchronization mechanisms can be seen as the 'Signed Echo Broadcast' implementation of a consistent broadcast on the label $(\mathsf{loc}_{BA}, \mathsf{Version})$ [20].

*Reading the store.* Steps ❽-❿ of Figure 4 illustrate the minimal interactions between user $A$ and the storage authorities to allow the user to read the distributed store. The user creates a *read transaction* to read the value $c_{BA}$ associated with a specified store entry $\mathsf{loc}_{BA} = g_1^{t_{BA}}$ (❽). Each authority replies with a *read reply* containing the latest value they hold for that store entry or None if the entry is not in their store (❾). Finally, user $A$ processes the replies performs the synchronization protocol described above (in case it did not terminate), and deduces the latest value associated with the queried key (❿). A detailed description of how readers process incoming read replies is included in the appendix of the extended paper.

[9]The extended version of the paper is available at https://eprint.iacr.org/2023/1218

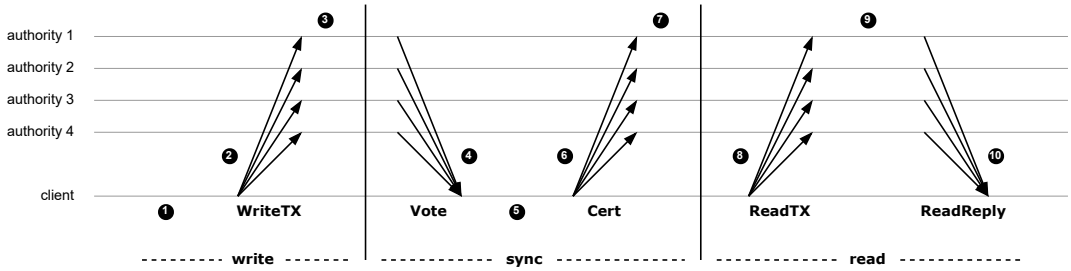


**Figure 4: Example of Arke write (❶-❸), sync (❹-❼), and read (❽-❿) protocol with 4 authorities.**

## 5.2 Existing Blockchains as Arke Store

Section 5.1 illustrates a minimal Arke store; we now show how Arke can natively leverage most types of existing blockchains as a store. User $A$ wishing to write the key-value pairs $(\mathsf{loc}_{AB}, c_{AB})$ to the store first format the key $\mathsf{loc}_{AB} = g_1^{t_{AB}}$ into a blockchain address $addr_{AB}$. Virtually all existing blockchains format public keys into addresses by hashing $addr_{AB} = H(g_1^{t_{AB}}||const)$, where $const$ is a public and blockchain-specific constant. The next paragraphs illustrate how to implement an Arke store over different types of blockchains.

*Payment-only platforms.* An Arke store can be any distributed payment platform where the transaction format allows user-defined metadata. For instance, Arke can easily use Bitcoin [58] as a store. A user $A$ wishing to write the store makes a Bitcoin transaction sending an arbitrary number of coins to the address $addr_{AB}$ (deterministically derived from $\mathsf{loc}_{AB}$ as mentioned above) and additionally, writes the `OP_RETURN` opcode. This opcode allows users to specify up to 80 arbitrary bytes within the transaction (by setting `OP_RETURN_MAX_BYTES` to 80); user $A$ writes the byte representation of $c_{AB}$. User $A$ reads the blockchains by locally generating $addr_{BA}$; it can then use any light client capable of parsing `OP_RETURN`, such as *Chain* [2], to retrieve the content of $addr_{BA}$ and parse $c_{BA}$. Alternatively, Arke can leverage other platforms not allowing to augment transactions with arbitrary metadata by encoding $c_{AB}$ in the less significant digits of the transfer amount.

*Smart contract platform.* An Arke store can also consist of any traditional smart contract platforms [9, 80] or rollup [5, 60]. A dedicated smart contract maintains a key-value map of the pairs $(\mathsf{loc}_{AB}, c_{AB})$ that users can easily read and write. To implement good state hygiene, both user $A$ and $B$ can delete an entry of the key-value map by proving knowledge of the secret key associated with $\mathsf{loc}_{AB}$ (locally derived).

*Leverage consensus-less operations.* Recent blockchains such as Sui [57] and Linera [3] allow users to program some types of transactions to entirely forgo consensus. For instance, Sui [57] is a smart-contract platform that forgoes consensus for single-writer operations and only relies on consensus for multi-writer operations, combining the two modes securely. As a result, any operation that can be expressed as a single-writer operation can leverage its consensus-less path and benefit from sub-second latency and lower gas fees. Arke can natively benefit from this feature. User $A$ writes the store by creating a *owned object* [14] containing $c_{AB}$ as the only field; it then transfers ownership of that object to the address $addr_{AB}$. User $A$ reads the blockchain by locally deriving $addr_{BA}$ and querying all objects owned by that address. The extended version of the paper also implements an Arke store on Sui using exclusively owned objects in less than 10 lines-of-code.

# 6 Implementation and Evaluation

We implement Arke's cryptographic operations in Rust, using the arkworks ecosystem [6]. The ID-NIKE is instantiated over the pairing group BLS12-377. The zkSNARK for $\mathcal{R}_{\mathsf{domID}}$ is instantiated by the Groth16 [42] proof system over the BW6-761 pairing group, in order to efficiently prove statements about variables from BLS12-377 [48]. The unlinkable handshake is implemented using Blake2X [8] as a key-derivation function and AES-GCM [38] with 256 bits blocks as the AEAD scheme.

We additionally implement and evaluate our custom Arke store described in Section 5.1. We open-source all our implementations[10] and measurement data to enable reproducible results[11]. In the following, we use `m5d.8xlarge` instances whenever experimenting on Amazon Web Services (AWS). These instances provide 10 Gbps of bandwidth, 32 virtual CPUs (16 physical cores) on a 2.5 GHz, Intel Xeon Platinum 8175, 128 GB memory, and run Linux Ubuntu server 22.04. We select this type of instance as it provides decent performance and is in the price range of 'commodity servers'.

## 6.1 Setup Phase

Table 1 shows the performance of all operations of the Arke setup protocol described in Section 4 on a single CPU core. We perform our benchmarks on both a `m5d.8xlarge` Amazon Web Services (AWS) instance and a Macbook Pro equipped with an M1 processor. The function *Assemble private key* is evaluated for a committee of 10 authorities. We compute the average time over 50 runs.

The table shows that user registration (performed by the registration authority) is cheap, taking respectively about 66 and 4 ms on our AWS instance and our M1 Macbook Pro. Generating private key requests is the most expensive operation; it takes about 23 seconds on our AWS instance and 2 seconds on an M1 Macbook Pro; this operation is however performed by the user (and only once) and thus does not take resources away from the key authorities. Issuing blind partial keys over a key request (performed by the key authority) is also cheap; it takes about 350 ms on our AWS instance and 20 ms on our M1 Macbook Pro, mostly spent verifying the

[10] https://github.com/asonnino/arke
[11] https://github.com/asonnino/arke/tree/main/code/arke/results/results-main

**Table 1: Microbenchmark of the Arke setup functions on a `m5d.8xlarge` AWS instance and a Macbook Pro equipped with an M1 CPU. Each data point represents the average time (over 50 runs) in milliseconds required to evaluate the function. The function *Assemble private key* is evaluated for a committee of 10.**

| Function | AWS | MBP |
|---|---|---|
| (RA) User registration | 66.12 ms | 4.13 ms |
| (User) Private key request | 23,402.37 ms | 2,259.66 ms |
| (KA) Issue blind partial key | 358.05 ms | 20.78 ms |
| (User) Assemble private key | 584.91 ms | 41.85 ms |

user's key request. Assembling a quorum of blind partial keys into a full private key (performed by the user) takes about 600 ms on our AWS instance and 41 ms on our M1 Macbook Pro. We implement this operation pessimistically requiring the user to verify each blind partial key before aggregation.

## 6.2 The Arke Custom Store

We implement a networked multi-core Arke store authority as described in Section 5.1. It uses `tokio`[12] for asynchronous networking and persists data structures using `Rocksdb`[13]. Our implementation uses TCP to achieve reliable point-to-point channels, necessary to correctly implement the distributed system abstractions. All operations use sha-256 for hashing and a simple DL proof over the curve BW6-761 to authenticate write requests as described in Section 5. We persist signed write requests before sending them (before step 4 of Figure 4) to ensure crash recovery. We store any other data asynchronously and out of the critical path to ensure that the sync protocol does not block on storage.

We particularly aim to demonstrate the performance claims of Section 2.3, reformulated as follows. **(C1)** Arke scales well with the committee size. **(C2)** Arke achieves low latency even under high load, in the WAN, and with large committee sizes. **(C3)** Arke achieves enough throughput to operate at planetary scale. **(C4)** Arke is robust when some parts of the system inevitably crash-fail. Note that evaluating BFT protocols in the presence of Byzantine faults is still an open question [12].

*Experimental setup.* We deploy a Arke testbed on AWS, using `m5d.8xlarge` instances across 10 different AWS regions: N. Virginia (us-east-1), Oregon (us-west-2), Canada (ca-central-1), Frankfurt (eu-central-1), Ireland (eu-west-1), London (eu-west-2), Mumbai (ap-south-1), Singapore (ap-southeast-1), Tokyo (ap-northeast-1), and Sydney (ap-southeast-2). All data are persisted on the NVMe drives provided by the AWS instance (rather than the root partition).

In the following graphs, each data point in the latency graphs is the average of the latency of all operations of the run, and the error bars represent one standard deviation (error bars are sometimes too small to be visible on the graph). We instantiate one benchmark client colocate with each authority submitting client requests at a fixed rate for 3 minutes. We benchmark two operations; (i) *write* and (ii) *write* followed by *synchronize* (see Section 5.1); we do not benchmark *read* as it is a simple database query common to many classic systems. When referring to *latency*, we mean the time elapsed from when the client submits the write request to when it assembles a certificate over the request (resp. when it is notified that a quorum of authorities is synchronized).

[12]https://github.com/tokio-rs/tokio
[13]https://rocksdb.org/

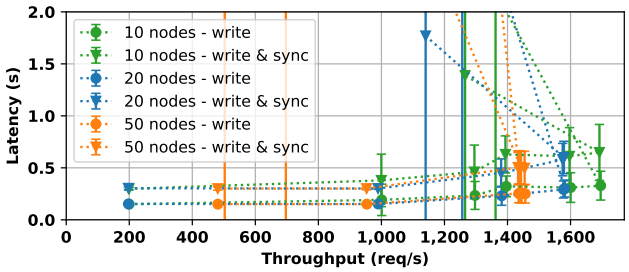


**Figure 5: Arke WAN latency-throughput with 10, 20, and 50 authorities (no faults); one shard per authority.**

We however note that clients wishing to retain unlinkability also at the network level would need to use an overlay network such as Tor [77], Nym [36], Apple Private Relay [4], or a distributed multi-hop VPN to access the store. This would add a constant overhead for generating new network identities and the latency of the chosen overlay network to the latency of the store, typically ranging from a few hundred milliseconds [78] to a couple of seconds [66, 67].

*Benchmark in the common case.* Figure 5 illustrates the latency-throughput of Arke for varying numbers of authorities. Every authority runs one shard (it thus runs on a single machine). We observe virtually no performance difference between runs with 10, 20, or even 50 authorities, thus validating our claim **(C1)**. Arke can process about 1,500 req/s with sub-second latency in all configurations. As expected the difference between simple *write* requests and *write* followed by *synchronize* is minimal. The latter displays a slightly higher latency due to the extra round-trip required to synchronize the authorities (about 100-200 ms) but throughput remains the same. This observation validates our claim **(C2)**. From the system usage estimates for the large-scale end-to-end encrypted messaging service WhatsApp (Section 1), we estimate the requirement to process around 120 req/s. Thus Arke exceeds by over 10x the throughput required to operate at this scale which validates claim **(C3)**. Even assuming that Facebook Messenger, Signal, and Telegram have similar usage to WhatsApp, Arke can process the combined load of these services and thus operate at a planetary scale.

*Benchmark under faults.* Figure 6 shows the performance of Arke for a 10-authorities deployment when the system is experiencing (crash-)faults; after running without faults for one minute, 0, 1, and 3 authorities permanently crash. Every authority runs a single shard (each authority thus runs on a single machine). The figure shows that there is no noticeable throughput drop under crash faults. Arke can finalize around 1,500 req/s with a sub-second latency. The latency slightly increases with the number of faulty authorities (by at most 200 ms). Clients finalize operations as soon as the fastest quorum of authorities replies (see Section 5.1); as authorities crash, clients are thus left with fewer authority replies from which to

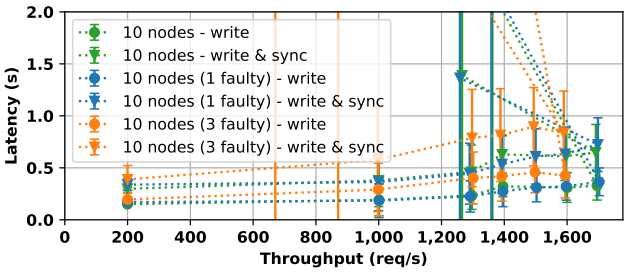


**Figure 6: Arke WAN latency-throughput with 10 validators (0, 1, and 3 faults); one shard per authority**

assemble certificates. This observation validates our claim **(C4)**. The performance of Arke shines compared to traditional consensus systems [13, 18, 19, 22, 29, 82] that are known to suffer 10x or 20x performance drop when experiencing leader failures [11, 18, 32, 45, 59, 68].

## 7 Related Work

In the following section we review related work and focus, in particular, on other contact discovery schemes. Additionally, we review works related to Arke's cryptographic primitives and Arke's store design in the extended version of the paper[14].

Arke implements a private contact discovery scheme by combining a key exchange with an unlinkable handshake [47]. This architecture generalizes the construction of Chaum *et al.* [26]. Their construction, known as UDM, implements both the key exchange and handshake by relying on honest-but-curious centralized parties. Furthermore, it requires to maintain a public mapping from (hashed) identifiers to public keys. Such a mapping reveals a public list of all registered users and requires storage that grows linearly in the number of system users. Finally, Chaum *et al.*[26] do not give proofs for the security and anonymity properties of their system.

Alternative architectures usually rely on Private Set Intersection (PSI). These protocols are particularly suited for the case of *mobile* contact discovery: when users' identifiers and messages are both their mobile phone numbers. Unfortunately, all PSI-based contact discovery schemes are vulnerable to enumeration attacks [44, 46, 49]. Indeed, even in its ideal functionality, PSI does not impose restrictions on what users present as being "their contacts" [31]. Therefore, a malicious client can enumerate the list of all other users by engaging in the PSI protocol honestly. This attack is described in the context of contact discovery by Hagen *et al.* [44], who show that mitigation strategies such as rate-limiting are not effective. Furthermore, PSI implies the existence of a centralized party that knows the list of all users and may act as a single point of failure. Whether this party can be distributed or thresholdized is an open problem which, to the best of our knowledge, has not been addressed to date. Arke mitigates both concerns by not requiring the set of all users to exist in a single location, and enforcing only bidirectional friendship relations.

Nonetheless, previous state-of-the-art mobile contact discovery schemes rely on PSI. Kiss *et al.* [50] introduce the notion of unbalanced PSI with precomputation. These PSI protocols are specifically tailored to the setting where one input set (the list of all users) is much larger than the other (a user's contacts). The computational and communication costs can also be split over three phases: the base phase, independent of either input set; the setup phase, depending only on the larger set; and the online phase, which can be made sublinear in the size of the larger set. Thus, a server that holds the list of all users can perform the base and setup phases once (for a fixed list of users) and answers user queries by only running the cheaper online phase. This approach is further refined by Kales *et al.* [49], who improve some of the cryptographic building blocks, and Hetz *et al.* [46], who introduce a private information retrieval (PIR) scheme to strike a trade-off in the communication costs of the setup and online phases. Demmler *et al.* [35] construct a contact discovery scheme from PIR and (balanced) PSI, assuming at least two non-colluding servers. Their approach does not allow for preprocessing and therefore requires the server to perform work that is linear in the number of users, for each query. We compare the asymptotic communication costs of PSI-based contact discovery schemes with those of Arke in Table 2.

**Table 2: Comparing communication cost of various PSI-based contact discovery schemes using PIR-PSI (by Demmler *et al.* [35]), unbalanced PSI (by Kales *et al.* [49]), and unbalanced PSI with PIR (by Hetz *et al.* [46]) to Arke's approach combining $(t, n)$-threshold oblivious ID-NIKE with an unlinkable handshake. Here $N$ denotes the total number of users, $c$ the number of a user's contacts, $m$ the number of partitions of the user set, and $t$ and $n$ threshold cryptography parameters.**

| Technique | Setup | Online |
|---|---|---|
| Demmler *et al.* [35] | – | $O(c \cdot \log(N/(c \log c)))$ |
| Kales *et al.* [49] | $O(N)$ | $O(c)$ |
| Hetz *et al.* [46] | $O(m\sqrt{N/m})$ | $O(c + \sqrt{c \cdot m \log m} \cdot \log(N/m))$ |
| This work | $O(t)$ | $O(c)$ |

Recently several state-of-the-art PSI protocols relying on oblivious transfer extension (OTe) [52, 63, 65, 69], oblivious key-value stores (OKVS) [64], or vector oblivious linear evaluation (VOLE) [40, 70] have been proposed but none of them have been used in the mobile contact discovery setting to the best of our knowledge. Moreover, although these schemes are the fastest for balanced PSI, they have not been studied under the lens of unbalanced PSI with precomputation. As a result, naively using these protocols in a contact discovery scheme will likely yield communication costs in the online phase that depend on the size of the set of users. Concurrent works published as pre-prints propose new techniques for unbalanced PSI [53, 74]. We leave the evaluation of these protocols and integration into contact discovery schemes as future work.

Signal [56] mimics the functionality of PSI using trusted hardware (Intel Software Guard Extensions (SGX)) and hides memory access patterns using Path ORAM [28]. This approach scales linearly in the number of users and suffers from the same privacy and fault-tolerance issues as PSI-based contact discovery. Furthermore, relying on Intel SGX requires trust in Intel [7, 33].

Pudding [51] (concurrent work) is an interactive protocol design to allow whistleblower to contact journalists in a privacy-preserving way. It is specifically designed to integrate with Nym [36]

[14]The extended version of the paper is available at https://eprint.iacr.org/2023/1218

and thus provides network-level anonymity. Pudding specifies the user registration protocol but relies on a set of pre-selected discovery nodes that drive the protocol by learning the metadata associated with all user queries. Arke can naturally interface with the user registration protocol of Pudding and leverage it to provide network-level anonymity. Furthermore, Arke is a general-purpose non-interactive bi-directional contact discovery system. There are thus no discovery nodes learning metadata associated with user queries. Finally, zkLogin [10] (concurrent work) is a privacy-preserving authentication protocol that allows users to derive a blockchain address from credentials derived from an OpenID Connect provider [39]. zkLogin does not natively implement contact discovery but can be leveraged to support generic openID connect providers as registration authorities (Section 2.1). We leave the exploration of combining Arke with zkLogin for future work.

## 8 Conclusion

Arke is the first Byzantine fault tolerant privacy-preserving contact discovery system whose performance is independent of the total number of users in the system (i.e., the *database size*). Our experimental implementation shows that Arke can support 1,500 user requests per second in a large geo-replicated environment, thus largely surpassing the combined estimated needs of WhatsApp, Facebook Messenger, Signal, and Telegram. Furthermore, Arke can maintain this throughput while providing sub-second finality even when a third of the infrastructure is Byzantine. Arke is based on an unlinkable handshake mechanism built on an ID-NIKE protocol and on a custom broadcast-based distributed architecture forgoing the expense of consensus.

## Acknowledgments

This work is partially supported by Mysten Labs and Geometry Research. We thank the cLabs team and the Celo community who inspired us to study privacy-preserving contact discovery several years ago. We thank Kostas Chalkias, Ben Riva, Joy, Francois Garillot, and Ge Gao for feedback on the early manuscript. Special thanks to Damir Shamanaev for suggesting the event-based implementation of the Arke smart contract-based store for Sui.

## References

[1] 2022. Telegram: A New Era of Messaging. https://telegram.org.
[2] 2023. The Bridge Between your Business and Web3. https://chain.com.
[3] 2023. Linera. https://linera.io.
[4] Apple. 2024. iCloud Private Relay. https://support.apple.com/en-us/102602.
[5] Arbitrum. 2023. Secure Scaling for Ethereum. https://arbitrum.io.
[6] arkworks contributors. 2022. `arkworks` *zkSNARK ecosystem.* https://arkworks.rs
[7] JP Aumasson and Luis Merino. 2016. SGX Secure Enclaves in Practice Security and Crypto Review. BlackHat.
[8] Jean-Philippe Aumasson, Samuel Neves, Zooko Wilcox-O'Hearn, and Christian Winnerlein. 2016. BLAKE2X. Online. https://www.blake2.net/blake2x.pdf
[9] Avalanche. 2023. Avalanche. https://www.avax.network.
[10] Foteini Baldimtsi, Konstantinos Kryptos Chalkias, Yan Ji, Jonas Lindstrøm, Deepak Maram, Ben Riva, Arnab Roy, Mahdi Sedaghat, and Joy Wang. 2024. zkLogin: Privacy-Preserving Blockchain Authentication with Existing Credentials. *arXiv preprint arXiv:2401.11735* (2024).
[11] Arati Baliga, I Subhod, Pandurang Kamat, and Siddhartha Chatterjee. 2018. Performance evaluation of the quorum blockchain platform. *ArXiv Preprint* (2018).
[12] Shehar Bano, Alberto Sonnino, Andrey Chursin, Dmitri Perelman, Zekun Li, Avery Ching, and Dahlia Malkhi. 2021. Twins: BFT Systems Made Robust. In *25th International Conference on Principles of Distributed Systems, OPODIS 2021, December 13-15, 2021, Strasbourg, France (LIPIcs, Vol. 217)*, Quentin Bramas, Vincent Gramoli, and Alessia Milani (Eds.). Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 7:1–7:29. https://doi.org/10.4230/LIPICS.OPODIS.2021.7
[13] Mathieu Baudet, Avery Ching, Andrey Chursin, George Danezis, François Garillot, Zekun Li, Dahlia Malkhi, Oded Naor, Dmitri Perelman, and Alberto Sonnino. 2019. State machine replication in the Libra Blockchain.
[14] Same Blackshear, Andrey Chursin, George Danezis, Anastasios Kichidis, Lefteris Kokoris-Kogias, Xun Li, Mark Logan, Ashok Menon, Todd Nowacki, Alberto Sonnino, Brandon Williams, and Lu Zhang. 2023. Sui Lutris: A Blockchain Combining Broadcast and Consensus. ArXiv Preprint.
[15] Alexandra Boldyreva. 2003. Threshold Signatures, Multisignatures and Blind Signatures Based on the Gap-Diffie-Hellman-Group Signature Scheme. In *Public Key Cryptography - PKC 2003, 6th International Workshop on Theory and Practice in Public Key Cryptography, Miami, FL, USA, January 6-8, 2003, Proceedings (Lecture Notes in Computer Science, Vol. 2567)*, Yvo Desmedt (Ed.). Springer, 31–46. https://doi.org/10.1007/3-540-36288-6_3
[16] Dan Boneh and Matt Franklin. 2001. Identity-based encryption from the Weil pairing. In *Crypto*.
[17] Dan Boneh, Ben Lynn, and Hovav Shacham. 2001. Short signatures from the Weil pairing. In *AsiaCrypt*.
[18] Ethan Buchman. 2016. *Tendermint: Byzantine fault tolerance in the age of blockchains.* Ph. D. Dissertation.
[19] Christian Cachin. 2016. Architecture of the hyperledger blockchain fabric. In *DCCL*.
[20] Christian Cachin, Rachid Guerraoui, and Luís Rodrigues. 2011. *Introduction to reliable and secure distributed programming.* Springer.
[21] Jan Camenisch, Susan Hohenberger, Markulf Kohlweiss, Anna Lysyanskaya, and Mira Meyerovich. 2006. How to Win the Clone Wars: Efficient Periodic n-Times Anonymous Authentication. In *CCS*.
[22] Miguel Castro and Barbara Liskov. 2002. Practical byzantine fault tolerance and proactive recovery. In *ACM Trans. Comput. Syst.*
[23] Laura Ceci. 2023. Monthly global unique WhatsApp users. https://www.statista.com/statistics/1306022/whatsapp-global-unique-users.
[24] Sofía Celi, Alex Davidson, Hamed Haddadi, Gonçalo Pestana, and Joe Rowell. 2023. Distefano: Decentralized infrastructure for sharing trusted encrypted facts and nothing more. ePrint Archive.
[25] Kwan Yin Chan, Handong Cui, and Tsz Hon Yuen. 2023. DIDO: Data Provenance from Restricted TLS 1.3 Websites. *ePrint* (2023).
[26] David Chaum, Mario Yaksetig, Alan T. Sherman, and Joeri De Ruiter. 2022. UDM: Private User Discovery with Minimal Information Disclosure. *Cryptologia* 46, 4 (July 2022), 347–379. https://doi.org/10.1080/01611194.2021.1911876
[27] Sherman SM Chow. 2009. Removing Escrow from Identity-Based Encryption: New Security Notions and Key Management Techniques. In *International workshop on public key cryptography*.
[28] Graeme Connell. 2022. Technology Deep Dive: Building a Faster ORAM Layer for Enclaves. https://signal.org/blog/building-faster-oram/.
[29] Corda. 2016. https://corda.net.
[30] Jean-Sébastien Coron, Jacques Patarin, and Yannick Seurin. 2008. The Random Oracle Model and the Ideal Cipher Model Are Equivalent. In *Advances in Cryptology - CRYPTO 2008, 28th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 17-21, 2008. Proceedings (Lecture Notes in Computer Science, Vol. 5157)*, David A. Wagner (Ed.). Springer, 1–20. https://doi.org/10.1007/978-3-540-85174-5_1
[31] Emiliano De Cristofaro, Mark Manulis, and Bertram Poettering. 2011. Private Discovery of Common Social Contacts. In *Applied Cryptography and Network Security - 9th International Conference, ACNS 2011, Nerja, Spain, June 7-10, 2011. Proceedings (Lecture Notes in Computer Science, Vol. 6715)*, Javier López and Gene Tsudik (Eds.). 147–165. https://doi.org/10.1007/978-3-642-21554-4_9
[32] George Danezis, Lefteris Kokoris-Kogias, Alberto Sonnino, and Alexander Spiegelman. 2022. Narwhal and Tusk: a DAG-based mempool and efficient BFT consensus. In *EuroSys*.
[33] Shaun Davenport. 2014. SGX: the good, the bad and the downright ugly. https://www.virusbulletin.com/virusbulletin/2014/01/sgx-good-bad-and-downright-ugly.
[34] Alex Davidson, Ian Goldberg, Nick Sullivan, George Tankersley, and Filippo Valsorda. 2018. Privacy Pass: Bypassing Internet Challenges Anonymously. In *PoPETS*.
[35] Daniel Demmler, Peter Rindal, Mike Rosulek, and Ni Trieu. 2018. PIR-PSI: scaling private contact discovery. *ePrint* (2018).
[36] Claudia Diaz, Harry Halpin, and Aggelos Kiayias. 2021. The Nym Network. (2021).
[37] Régis Dupont and Andreas Enge. 2006. Provably secure non-interactive key distribution based on pairings. *Discrete Applied Mathematics* 154, 2 (2006), 270–276. https://doi.org/10.1016/j.dam.2005.03.024 Coding and Cryptography.
[38] M J Dworkin. 2007. Recommendation for block cipher modes of operation: GaloisCounter Mode (GCM) and GMAC. https://doi.org/10.6028/nist.sp.800-38d
[39] OpenId Foundation. 2024. Certified OpenID Connect Implementations. https://openid.net/developers/certified-openid-connect-implementations/.
[40] Gayathri Garimella, Benny Pinkas, Mike Rosulek, Ni Trieu, and Avishay Yanai. 2021. Oblivious Key-Value Stores and Amplification for Private Set Intersection. In *Advances in Cryptology - CRYPTO 2021 - 41st Annual International Cryptology*

*Conference, CRYPTO 2021, Virtual Event, August 16-20, 2021, Proceedings, Part II (Lecture Notes in Computer Science, Vol. 12826)*, Tal Malkin and Chris Peikert (Eds.). Springer, 395–425. https://doi.org/10.1007/978-3-030-84245-1_14

[41] Rosario Gennaro, Stanisław Jarecki, Hugo Krawczyk, and Tal Rabin. 1999. Secure distributed key generation for discrete-log based cryptosystems. In *EuroCrypt*.

[42] Jens Groth. 2016. On the size of pairing-based non-interactive arguments. In *EuroCrypt*.

[43] Kobi Gurkan, Philipp Jovanovic, Mary Maller, Sarah Meiklejohn, Gilad Stern, and Alin Tomescu. 2021. Aggregatable distributed key generation. In *EuroCrypt*.

[44] Christoph Hagen, Christian Weinert, Christoph Sendner, Alexandra Dmitrienko, and Thomas Schneider. 2020. All the Numbers are US: Large-scale Abuse of Contact Discovery in Mobile Messengers. ePrint Archive.

[45] Runchao Han, Gary Shapiro, Vincent Gramoli, and Xiwei Xu. 2019. On the performance of distributed ledgers for internet of things. In *Internet of Things*.

[46] Laura Hetz, Thomas Schneider, and Christian Weinert. 2023. Scaling Mobile Private Contact Discovery to Billions of Users. *ePrint* (2023).

[47] Jaap-Henk Hoepman. 2015. Privately (and unlinkably) exchanging messages using a public bulletin board. In *Proceedings of the 14th ACM Workshop on Privacy in the Electronic Society*. 85–94.

[48] Youssef El Housni and Aurore Guillevic. 2020. Optimized and Secure Pairing-Friendly Elliptic Curves Suitable for One Layer Proof Composition. In *Cryptology and Network Security - 19th International Conference, CANS 2020, Vienna, Austria, December 14-16, 2020, Proceedings (Lecture Notes in Computer Science, Vol. 12579)*, Stephan Krenn, Haya Schulmann, and Serge Vaudenay (Eds.). Springer, 259–279. https://doi.org/10.1007/978-3-030-65411-5_13

[49] Daniel Kales, Christian Rechberger, Thomas Schneider, Matthias Senker, and Christian Weinert. 2019. Mobile Private Contact Discovery at Scale. In *USENIX*.

[50] Ágnes Kiss, Jian Liu, Thomas Schneider, N Asokan, and Benny Pinkas. 2017. Private set intersection for unequal set sizes with mobile applications. *ePrint* (2017).

[51] Ceren Kocaoğullar, Daniel Hugenroth, Martin Kleppmann, and Alastair R Beresford. 2023. Pudding: Private User Discovery in Anonymity Networks. *ArXiv Preprint* (2023).

[52] Vladimir Kolesnikov, Ranjit Kumaresan, Mike Rosulek, and Ni Trieu. 2016. Efficient Batched Oblivious PRF with Applications to Private Set Intersection. In *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security, Vienna, Austria, October 24-28, 2016*, Edgar R. Weippl, Stefan Katzenbeisser, Christopher Kruegel, Andrew C. Myers, and Shai Halevi (Eds.). ACM, 818–829. https://doi.org/10.1145/2976749.2978381

[53] Jörn Kußmaul, Matthew Akram, and Anselme Tueno. 2023. Unbalanced Private Set Intersection from Homomorphic Encryption and Nested Cuckoo Hashing. *IACR Cryptol. ePrint Arch.* (2023), 1670. https://eprint.iacr.org/2023/1670

[54] Jan Lauinger, Jens Ernstberger, Andreas Finkenzeller, and Sebastian Steinhorst. 2023. Janus: Fast Privacy-Preserving Data Provenance For TLS 1.3. *ePrint* (2023).

[55] Moxie Marlinspike. 2014. The Difficulty of Private Contact Discovery.

[56] Moxie Marlinspike. 2017. Technology Preview: Private Contact Discovery for Signal. https://signal.org/blog/private-contact-discovery.

[57] Mysten Labs. 2022. Sui: Build without Boundaries. https://sui.io.

[58] Satoshi Nakamoto. 2008. Bitcoin: A peer-to-peer electronic cash system.

[59] Qassim Nasir, Ilham A Qasse, Manar Abu Talib, and Ali Bou Nassif. 2018. Performance analysis of hyperledger fabric platforms. *Security and Communication Networks* 2018 (2018).

[60] Optimism. 2023. Ethereum, Scaled. https://www.optimism.io.

[61] Kenneth G Paterson and Sriramkrishnan Srinivasan. 2009. On the relations between non-interactive key distribution, identity-based encryption and trapdoor discrete log groups. In *Designs, Codes and Cryptography*.

[62] Torben Pryds Pedersen. 1991. A threshold cryptosystem without a trusted party. In *EuroCrypt*.

[63] Benny Pinkas, Mike Rosulek, Ni Trieu, and Avishay Yanai. 2019. SpOT-Light: Lightweight Private Set Intersection from Sparse OT Extension. In *Advances in Cryptology - CRYPTO 2019 - 39th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 18-22, 2019, Proceedings, Part III (Lecture Notes in Computer Science, Vol. 11694)*, Alexandra Boldyreva and Daniele Micciancio (Eds.). Springer, 401–431. https://doi.org/10.1007/978-3-030-26954-8_13

[64] Benny Pinkas, Mike Rosulek, Ni Trieu, and Avishay Yanai. 2020. PSI from PaXoS: Fast, Malicious Private Set Intersection. In *Advances in Cryptology - EUROCRYPT 2020 - 39th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Zagreb, Croatia, May 10-14, 2020, Proceedings, Part II (Lecture Notes in Computer Science, Vol. 12106)*, Anne Canteaut and Yuval Ishai (Eds.). Springer, 739–767. https://doi.org/10.1007/978-3-030-45724-2_25

[65] Benny Pinkas, Thomas Schneider, and Michael Zohner. 2014. Faster Private Set Intersection Based on OT Extension. In *Proceedings of the 23rd USENIX Security Symposium, San Diego, CA, USA, August 20-22, 2014*, Kevin Fu and Jaeyeon Jung (Eds.). USENIX Association, 797–812. https://www.usenix.org/conference/usenixsecurity14/technical-sessions/presentation/pinkas

[66] Tor Project. 2024. Tor Performance Metrics. https://metrics.torproject.org/onionperf-latencies.html.

[67] Tor Project. 2024. Tor Performance Metrics. https://metrics.torproject.org/onionperf-buildtimes.html.

[68] R3. 2020. *Sizing and Performance*. https://docs.corda.r3.com/sizing-and-performance.html

[69] Peter Rindal and Mike Rosulek. 2017. Malicious-Secure Private Set Intersection via Dual Execution. In *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security, CCS 2017, Dallas, TX, USA, October 30 - November 03, 2017*, Bhavani Thuraisingham, David Evans, Tal Malkin, and Dongyan Xu (Eds.). ACM, 1229–1242. https://doi.org/10.1145/3133956.3134044

[70] Peter Rindal and Phillipp Schoppmann. 2021. VOLE-PSI: Fast OPRF and Circuit-PSI from Vector-OLE. In *Advances in Cryptology - EUROCRYPT 2021 - 40th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Zagreb, Croatia, October 17-21, 2021, Proceedings, Part II (Lecture Notes in Computer Science, Vol. 12697)*, Anne Canteaut and François-Xavier Standaert (Eds.). Springer, 901–930. https://doi.org/10.1007/978-3-030-77886-6_31

[71] R Sakai, K Ohgishi, and M Kasahara. 2000. Cryptosystems based on pairing. In *The 2000 Symposium on Cryptography and Information Security*. 26–28.

[72] Signal. 2022. Signal: Speak Freely. https://signal.org.

[73] Ai-fen Sui, Sherman SM Chow, Lucas Chi Kwong Hui, Siu-Ming Yiu, Kam-Pui Chow, Wai Wan Tsang, CF Chong, KH Pun, and HW Chan. 2005. Separable and anonymous identity-based key issuing. In *11Th international conference on parallel and distributed systems (ICPADS'05)*, Vol. 2. IEEE, 275–279.

[74] Yunqing Sun, Jonathan Katz, Mariana Raykova, Phillipp Schoppmann, and Xiao Wang. 2024. Large-Scale Private Set Intersection in the Client-Server Setting. *IACR Cryptol. ePrint Arch.* (2024), 570. https://eprint.iacr.org/2024/570

[75] Telegram. 2022. Telegram Privacy Policy. https://telegram.org/privacy.

[76] TLSNotary. 2023. TLSNotary [source code]. https://github.com/tlsnotary/tlsn.

[77] Tor. 2023. Tor Project. https://www.torproject.org.

[78] Martino Trevisan, Idilio Drago, Paul Schmitt, and Francesco Bronzino. 2023. Measuring the performance of icloud private relay. In *International Conference on Passive and Active Network Measurement*. Springer, 3–17.

[79] WhatsApp LLC. 2022. WhatsApp: Simple, Secure, Reliable messaging. https://www.whatsapp.com.

[80] Gavin Wood et al. 2014. Ethereum: A secure decentralised generalised transaction ledger. Ethereum project yellow paper.

[81] Xiang Xie, Kang Yang, Xiao Wang, and Yu Yu. 2023. Lightweight Authentication of Web Data via Garble-Then-Prove. *ePrint* (2023).

[82] Maofan Yin, Dahlia Malkhi, Michael K Reiter, Guy Golan Gueta, and Ittai Abraham. 2019. Hotstuff: Bft consensus with linearity and responsiveness. In *PODC*.

[83] Fan Zhang, Deepak Maram, Harjasleen Malvai, Steven Goldfeder, and Ari Juels. 2020. Deco: Liberating web data using decentralized oracles for TLS. In *CCS*.