This is the final peer-reviewed accepted manuscript of:

Towards Next Generation Domain Services: A Committee based Privacy Scheme leveraging Move on IOTA

Conference Proceedings: 2025 7th Conference on Blockchain Research & Applications for Innovative Networks and Services (BRAINS), November 18-21, 2025, Zurich, Switzerland.

Author: Antonino Iaria; Mirko Zichichi; Vittorio Ghini; Stefano Ferretti

Publisher: IEEE

The final published version is available online at:
https://doi.org/10.1109/BRAINS67003.2025.11302929

Rights / License:

© 2025 IEEE. Personal use of this material is permitted. Permission from IEEE must be obtained for all other uses, in any current or future media, including reprinting/republishing this material for advertising or promotional purposes, creating new collective works, for resale or redistribution to servers or lists, or reuse of any copyrighted component of this work in other works. The terms and conditions for the reuse of this version of the manuscript are specified in the publishing policy. For all terms of use and more information see the publisher's website:

https://www.ieee.org/content/dam/ieee-org/ieee/web/org/pubs/author_version_faq.pdf

This item was downloaded from the author personal website (https://mirkozichichi.me)

When citing, please refer to the published version.

Towards Next Generation Domain Services: A Committee based Privacy Scheme leveraging Move on IOTA

1st Antonino Iaria
University of Bologna
Bologna, Italy
antonino.iaria3@studio.unibo.it

2nd Mirko Zichichi
IOTA Foundation
Berlin, Germany
mirko.zichichi@iota.org

3rd Vittorio Ghini
University of Bologna
Bologna, Italy
vittorio.ghini@unibo.it

4th Stefano Ferretti
University of Bologna
Bologna, Italy
s.ferretti@unibo.it

***Abstract*—The inherent complexity of cryptographic addresses in Web3 ecosystems poses significant usability problems. Decentralized Domain Services attempt to solve this challenge yet, in the meantime, jeopardize users' pseudo-anonymity. This paper investigates the design and implementation of a decentralized naming system leveraging IOTA's Move smart contract framework. We analyze the system architecture and provide a comparative evaluation against established domain name services such as ENS, SuiNS, SolanaNS, and Avvy, highlighting the unique advantages and limitations of our proposed solution. Moreover, we introduce a novel committee-bridge architecture managing off-chain hashed name storage, enabling selective disclosure of domain information while maintaining low computational overhead. We also present results from an evaluation that assesses the functioning and the performance of the system, which confirm the viability of the proposal.**

I. Introduction

Web3 domain services encompass diverse designs [1], yet most still reference accounts via *addresses* or *public keys*, obliging users to share non-human-readable strings and undermining usability, particularly for non-experts. *Account abstraction* seeks to streamline interaction by elevating smart contract wallets to first-class accounts and reducing reliance on externally owned accounts.

In this paper, we implement name services using IOTA's Move smart contract language [2], [3]. The address-naming process mirrors Internet DNS [4] but operates in a decentralized architecture; Figure 1 outlines our approach, which balances privacy, legal compliance, and transparency.

The expansion of name services faces scalability and congestion, notably in Ethereum [5], as adoption grows. Stakeholders are advancing capacity via off-chain mechanisms and regulatory compliance strategies; this work proposes an alternative that stores sensitive data hybridly on-chain while addressing current limitations.

We present a decentralized name service with a hybrid architecture combining on-chain transparency and off-chain confidentiality. A committee-bridge middleware enables stealth registrations by hashing user-submitted names and storing them off-chain; only authorized committee members can resolve them to plaintext. Public registrations remain fully on-chain. Access control follows a capability-based model implemented in IOTA Move: registrants obtain capabilities they can delegate to grant access to stealth domains, enabling secure, fine-grained, on-chain enforcement.

The remainder of this paper is organized as follows. Section 2 reviews background and related work; Section 3 introduces the proposed system; Section 4 details the implementation; Section 5 reports the performance evaluation; Section 6 concludes.

TABLE I
Comparison of Web3 NS by feature, ownership, and language.

Service	Web2	Privacy	Denylist	Ownership	Language
ENS	Yes	No	No	ERC-721 NFT	Solidity
SuiNS	No	No	Yes	Tokenized	Move
SolanaNS	No	No	No	Registry	Rust
Avvy	Yes	Yes	Yes	ERC-721 NFT	Solidity

II. Background

Today, Web3 space offers multiple on-chain naming systems with overlapping yet distinct designs; Table I compares key features across widely used DLTs. ENS on Ethereum [5] resolves human-readable names (e.g., 'alice.eth') to addresses, content hashes, and metadata. SuiNS on Sui [6], implemented in Move, currently lacks privacy controls. SolanaNS on Solana [7], written in Rust, prioritizes very low latency over privacy features. Avvy on Avalanche [8] targets identity and interoperability across subnets, with Solidity contracts supporting private metadata fields, namespaces, and modular, selective data disclosure.

A. Domain Registration and Ownership

ENS implements registration and resolution via Solidity smart contracts; .eth domains are represented as ERC-721 NFTs, with the ETH Registrar administering the TLD for trustless issuance [5]. Names are normalized and hashed to a node to reduce storage, while hierarchical subdomains and custom resolvers remain supported.

SuiNS on Sui, implemented in *Move* [2], organizes lookups and reverse via table collections with on-chain or off-chain resolution; ownership is tokenized, with currently limited user-data records (e.g., avatar, IPFS) [**?**].

The Solana Name Service enables .sol registrations via a DApp, managing ownership and records through Solana's account model and PDAs; subdomains are supported, and updates/transfers occur via the interface or wallet instructions.

Avvy on Avalanche's EVM-compatible C-Chain manages names via Solidity contracts with ERC-721 ownership and fixed-price or auction registrations; modular records cover addresses and metadata, while Enhanced Privacy and hashed/off-chain proofs provide selective disclosure, with some private-record features still emerging [8].

B. Access Control Policies

Access control in name services remains challenging, as many operate without explicit policies. ENS—the most widely used decentralized naming system—prioritizes open, universal resolution and broad integration, simplifying UX but constraining privacy and access control. Our project addresses this gap with granular, authorization-based name resolution to protect sensitive data and strengthen digital identity management. On Avalanche, Avvy similarly adds owner-controlled disclosure and user authorization.

C. IOTA and the Move Language

IOTA and Move enable the development of scalable, secure, and user-friendly decentralized digital ecosystems. IOTA provides a flexible and robust infrastructure built for real-world asset tokenization and decentralized application interactions, leveraging a DAG-based proof-of-stake consensus [9]. Its hybrid ledger model supports parallel transaction processing and horizontal scaling, allowing for unrestricted throughput and EVM-compatible chains that foster interoperable layer 2 ledgers. Meanwhile, Move brings an object-centric programming language specifically tailored for digital asset management. Its design eliminates common smart contract vulnerabilities through rigorous compile-time checks and an architecture based on isolated objects rather than a shared global state. Move draws inspiration from Rust's ownership model for memory management and resource control. Its compiler prevents common smart contract programming mistakes, ensuring that assets remain secure within user accounts and are only accessible with the proper keys.

D. Move Access Control

While ACLs are established in Web3, Move language offers a promising alternative. This paper examines Move's capabilities to address current limitations, contrasting Solidity's map-based user records with Move's resource-direct object assignment. The result is simpler, more efficient record resolution with lower gas fees. Table II shows a summary of differences.

Move leverages capability objects for flexible access control; these must be passed as function parameters to authorize privileged actions, enforcing least privilege at the type level and enabling fine-grained, verifiable control.

By contrast, Solidity relies on runtime modifiers checks over addresses or ownership which are easier and more familiar to implement but depend heavily on developer expertise.

III. System Design

The prototype is a middleware system that handles the mapping between stealth names (using cryptographic-secure hash-functions [**?**]) and blockchain addresses. Users register their name/domain through the bridge. When someone wants to resolve a stealth name, a query to the bridge system is done using the identifier from the blockchain. Some extensive ACL policies, or equivalent solutions, must be implemented to manage domain lookups.

In order to keep the discussion generic and not dependent on the type of bridge implementation, we will use the term "bridge's storage" to refer to the space where the bridge system records its data, whether it is a normal database (centralized solution), distributed storage (peer-to-peer system), or a ledger (permissioned blockchain). More details are given in Section III-D. Our implementation is based on Move's pattern *proof of authority* to manage access to hashed names. This solution is based on the use of capabilities, i.e., a security model based on possession of unforgeable tokens that grant specific permissions. In particular, we introduce *ParentCap* and *ChildCap*. The first capability is only issued to the owner of the stealth domain. ChildCap is given to any additional user from the owner of the ParentCap. The *stealth* feature relies on a cryptographic hash function. When a user submits a name to be hidden, the bridge applies a salt and computes its hash. The resulting digest is stored both off-chain in the bridge system, and on-chain for reference and reverse lookups. While the digest is publicly visible on the distributed ledger, the corresponding plain-text name remains private, known only to the designated committee. Although the owner of a stealth name can be identified on-chain, the original name itself cannot be derived, feasibly, from the available data. Final proposed system has been developed to do the following use cases:

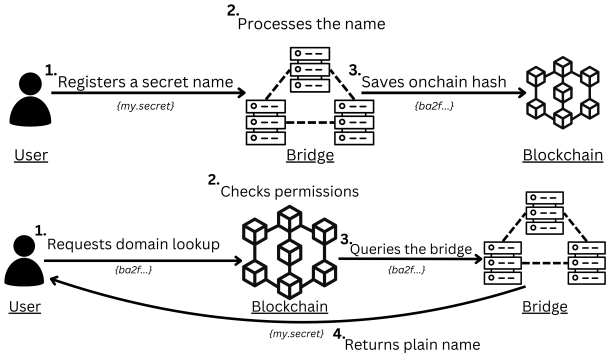


Fig. 1. This diagram illustrates the *stealth* privacy mechanism. For secret name registration (top), a user interacts with the bridge committee, which commits the necessary information on-chain. For lookup (bottom), user first queries blockchain and then requests committee to reveal plaintext name.

1) **UC1 Non-Stealth domain registration:** User selects a string to register, executes a transaction onchain;
2) **UC2 Stealth domain registration:** User selects a name to register. The bridge saves the tuple name-hash and then register it, hashed, on-chain.
3) **UC3 Non-Stealth domain lookup:** User enters a name.If the given name is found, on-chain, the system shows correspondent owner and NFT object Id.
4) **UC4 Stealth domain lookup:** User enters hashed name to be resolved and a Move capability (on-chain). If the given information is correct, the committee-bridge queries the plain name for the given hashed one returning it back.
5) **UC5 Add new user to the ACL**: An authorized user (owner of a Move capability object) grants permissions (as another Move capability) to others.

A. System's actors

Interactions are mainly between users. However, it is worth specifying all the involved actors of the platform, more in details:

- **Registrant User:** Registers a domain and manages AC.
- **Lookup User:** Requests domain access; for stealth, must be

TABLE II
COMPARISON BETWEEN MOVE AND SOLIDITY IN SMART CONTRACT DEVELOPMENT

	Move	Solidity
Resource Model	Resource-oriented, linear types prevent duplication or loss	Account-based model, traditional ownership semantics
State Storage	Data is structured under modules with fine-grained control	Uses contract storage with global mappings and variables
Run Model	Transaction scripts operate over modules; no global dispatcher	Function calls via public or external visibility; uses 'msg.sender'
Error Handling	Deterministic via 'abort' codes and strict runtime rules	Uses 'require', 'revert', 'assert'; error messages are optional
Composability	Strong type and module system; encourages explicit dependencies	Composable via interfaces/inheritance; complexity risk.

an authenticated capability holder (ParentCap or ChildCap) linked to the specific name record.

- **Committee Bridge Node:** Participates in the committee bridge and answers stealth-domain requests.
- **Blockchain Full Node:** Participates in the blockchain network, handling users' and committee nodes' transaction submissions and ledger reads.

B. System Components

The designed system has to be balanced in terms of scalability, data transparency and privacy of shared information. To this extent, the required system components are:

- **Committee Bridge:** The core of our system, i.e., a trusted middleware aimed at preserving privacy of information.
- **Storage:** Depending on the implementation of the committee bridge, an efficient data structure is needed.
- **Blockchain networks:** A suitable DLT should enable efficient name registration with low fees; for legal compliance, a permissioned solution is preferable.

C. Data Sharing

Since the proposed system must have both on-chain and off-chain components, it is important to design appropriate ways to handle interactions and share data across different networks.

1) **UC1[On-Chain]:** Name & metadata are registered and stored on-chain.
2) **UC2[Hybrid]:** Name is off-chain; metadata is on-chain.
3) **UC3[On-Chain]:** Public data is resolved directly via smart contract queries.
4) **UC4[Hybrid]:** On-chain capability authorizes off-chain name reveal.
5) **UC5[On-Chain]:** Capabilities are updated via smart contract calls.

D. Committee Bridge

The Committee Bridge is a decentralized, secure interoperability protocol in which a validator committee verifies and propagates cross-domain messages in a trust-minimized way. A robust design employs BFT consensus among committee, PBFT [10] or related mechanisms [11], [12] with dynamic membership via reputation or decentralized governance. To achieve high performance, it must scale across domains and validator sets, sustaining higher throughput and low latency; DAG-based schemes can reduce validator load while addressing consensus, scalability, security, and extensibility.

IV. IMPLEMENTATION

The following sections describe the main components and technologies used to build the prototype.

- **Smart contracts:** 5 Move language modules, deployed on the IOTA developer network under the same package.
- **Committee Bridge:** Developed using IOTA's Rust SDK. It acts as trusted middleware for the operations between users and smart contracts.
- **Bridge Storage:** For the purpose of this investigation, it is implemented as a lightweight JSON store that serves as a simple key-value repository for storing tuples.

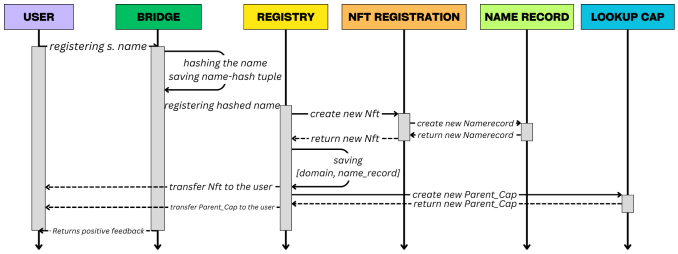


Fig. 2. Sequence Diagram for registering a **stealth** domain

V. PERFORMANCE EVALUATION

We evaluate the feasibility of a Rust-based blockchain bridge that mimics DNS functionality while preserving name privacy. The core goal is to assess the performance and behavior of the system under concurrent client load in a controlled, reproducible environment.

A. Design

We decided to mainly focus on the UC2 (registering a stealth name). This operation requires different agents and concurrent resource access. For these reasons, we believe it might represent a fair comparison use case to the others, that we expect to be less computational and latency demanding compared to the analyzed one. The bridge is implemented as a single Rust program that acts as a local name-resolution system with stealth features. The bridge receives plaintext identifiers (names) from clients, computes a cryptographic hash for each, and stores both the plaintext and hash locally as a tuple. Only the hashed representation is published on-chain, preserving the stealth aspect.

B. Methodology

To simulate realistic usage and enable metric collection, the entire testing setup was containerized using docker compose. The key components include:

- **Bridge Container:** Runs Rust logic to hash the name, store the digest–plaintext tuple, and call the smart contract with the digest.
- **Agent Containers:** Concurrently send name-resolution requests to the bridge; each client emulates an independent user.
- **Dashboard Container:** Python service visualizing metrics aggregated from all clients.

All containers communicate on a shared Docker network. The distributed bridge listens on port 5000. For this experiment, authentication is not enforced and is assumed to occur beforehand.

C. Metrics, data Collection and Results

The analyzed latency includes the following steps:

1) Agents send an HTTP POST to the bridge (address:port) with the name to hash, the claimant address, and a request number to prevent gas coin reuse (see Section V-D).
2) The bridge receives and validates the request.
3) Computes a Blake2 digest.
4) Stores the digest locally alongside the plain name.
5) Calls the smart contract's *add_record*.
6) Loads agent coins and selects one to pay gas fees.
7) Signs the transaction with the agent keypair and submits on-chain; the DLT validates and updates state.
8) Returns the registration outcome as an HTTP status code.

We evaluated five load scenarios with 1–50 concurrent users; each issued 100 sequential requests to approximate real-world traffic. Figure 3 reports latencies at 50 concurrent users: spikes approach 2 s, but most requests complete quickly. Figure 4 summarizes statistics across workloads, where higher concurrency increases standard deviation while average latency remains low. In the 50-agent run, 0,04% of requests were dropped. Overall, the system shows promising performance under load.

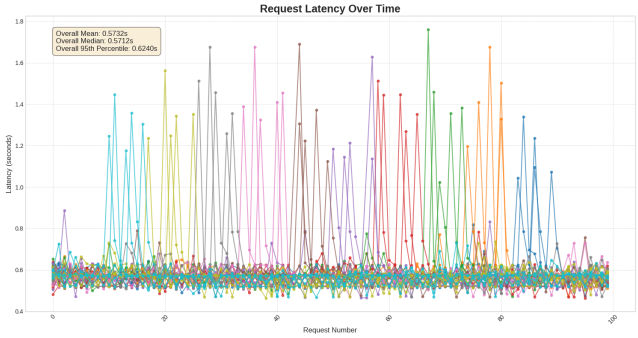


Fig. 3. Latencies for 5000 total requests by 50 concurrent agents

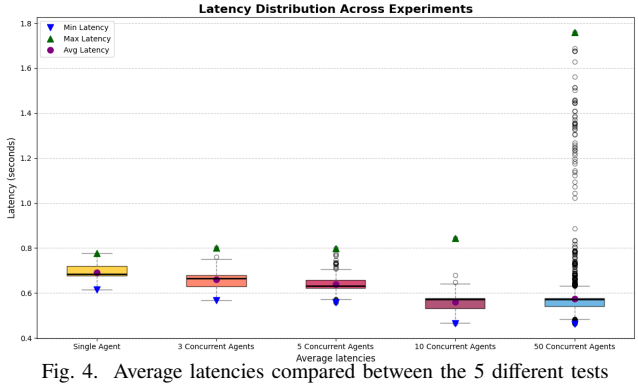


Fig. 4. Average latencies compared between the 5 different tests

D. Discussion

The initial setup was a critical aspect of the experiment. Although the IOTA consensus mechanism is capable of handling a high transaction rate per second, each transaction needs a unique gas object to be fulfilled.

This design effectively prevents double-spending attacks on one side, but on the other, introduces the requirement for a robust data structure to ensure the same coin is not used more than once during stress tests.

Lastly, gas fees should also be something to keep in mind. IOTA's mechanisms is similar to EVM, where a computational and storage cost is calculated and payed in order to execute such transactions. The delicate topic indeed, is the expensive costs related to long, verbose, names. On the other hand, stealth names have a fixed-digest length which was helpful to predetermine the gas budget for each transaction. For the conducted tests, we capped the gas budget to 8M NANOs (0.008 IOTA) having all the stealth name registered using a little less than 7M.

VI. Conclusions

This paper implements capability-based access control for decentralized name services in Move on IOTA. The ParentCap and ChildCap primitives provide tunable, verifiable control over stealth-domain resolution, restricting reverse lookups to authorized parties. This marks an initial phase of a privacy-preserving, flexible naming infrastructure that strengthens access management and user control over existing systems.

Regarding fees, Solidity/Ethereum supports gas-free, view-only calls for non–state-changing functions, whereas Move currently lacks an equivalent. The prevailing practice is a dry run that validates a transaction without committing, enabling gas-less capability verification but relying on validator trust.

Future work should replace the single bridge with a distributed, fault-tolerant committee. Access control must add expiration, dynamic revocation, and automated cleanup to support complex policies, and incorporate richer personal data beyond hashed names.

While the current system is generic, future iterations should be designed for specific use cases. Public administration is especially promising, where secure, privacy-preserving, user-controlled identity systems are highly relevant. By refining use cases around digital identity, privacy management, and transparent procurement, the system can deliver real value.

References

[1] C. Chen, L. Zhang, Y. Li, T. Liao, S. Zhao, Z. Zheng, H. Huang, and J. Wu, ''When digital economy meets web3. 0: Applications and challenges,'' *IEEE Open Journal of the Computer Society*, vol. 3, pp. 233–245, 2022.

[2] S. Blackshear, E. Cheng, D. L. Dill, V. Gao, B. Maurer, T. Nowacki, A. Pott, S. Qadeer, D. R. Rain, S. Sezer *et al.*, ''Move: A language with programmable resources,'' *Libra Assoc*, vol. 1, 2019.

[3] S. Popov, ''The tangle,'' *White paper*, vol. 1, no. 3, p. 30, 2018.

[4] P. Mockapetris and K. J. Dunlap, ''Development of the domain name system,'' in *Symposium proceedings on Communications architectures and protocols*, 1988, pp. 123–133.

[5] V. Buterin *et al.*, ''Ethereum whitepaper,'' *GitHub repository*, vol. 1, no. 22-23, pp. 5–7, 2013.

[6] ''The sui smart contracts platform,'' https://docs.sui.io/paper/sui.pdf, [Accessed 25-04-2025].

[7] A. Yakovenko, ''Solana: A new architecture for a high performance blockchain v0. 8.13,'' 2018.

[8] ''Avalanche whitepapers,'' https://www.avalabs.org/whitepapers, [Accessed 25-04-2025].

[9] N. Polyanskii, S. Mueller, and I. Vorobyev, ''Starfish: A high throughput BFT protocol on uncertified DAG with linear amortized communication complexity,'' Cryptology ePrint Archive, Paper 2025/567, 2025. [Online]. Available: https://eprint.iacr.org/2025/567

[10] M. Castro, B. Liskov *et al.*, ''Practical byzantine fault tolerance,'' in *OsDI*, vol. 99, no. 1999, 1999, pp. 173–186.

[11] H. Moniz, ''The istanbul bft consensus algorithm,'' *arXiv preprint arXiv:2002.03613*, 2020.

[12] D. Mazieres, ''The stellar consensus protocol: A federated model for internet-level consensus,'' *Stellar Development Foundation*, vol. 32, no. 4, pp. 1–45, 2015.