Blockchain's Role in Metaverse Trust and Transactions

R. Can Aygun
UCLA
Los Angeles, USA
Email: rcaygun@cs.ucla.edu

Turan Vural
UCLA
Los Angeles, USA
Email: turan@cs.ucla.edu

Lixia Zhang
UCLA
Los Angeles, USA
Email: lixia@cs.ucla.edu

***Abstract*—The metaverse is a new technology which promises to enhance human-to-computer interaction and achieve an immersive experience in computing and communication. In parallel, blockchain is viewed by many as a pillar of trust in creating a secure and decentralized metaverse. In this paper, we first articulate the functionalities required by a simple metaverse use case, and then examine the challenges an strengths of a blockchain-based implementation in providing those functions. Our analysis shows that blockchain-based solutions may require centralized functions in name lookup, lacks assurance for name uniqueness and accessible data authentication as of now, and faces scalability challenges due to the high demand on computing and storage resources.**

1. Introduction

The metaverse has gained attention from industry [1] and academic communities [2] in recent years as a promising technology to achieve immersive computing and communication capabilities. It has come to reference an intersection of different technologies: AI, AR/VR, blockchain, 3D rendering, IoT, and edge computing [3]. Among them, blockchains, especially in its capacity as a distributed ledger, have been deemed by some prominent studies and industry as the pillar of trust and transactions to create a secure metaverse [1], [4]. For example, Lamina1 is a blockchain that is aimed to be an underlying infrastructure for data storage, identity, messaging and payments in the metaverse [4].

In this paper, we use a simple metaverse application scenario to perform a preliminary investigation into anonymous blockchain's role in supporting trust policies and transaction recording in an open metaverse[1]. We use Ethereum, one realization of blockchain technology, to investigate blockchain's capability in supporting metaverse applications and identify potential challenges and limitations in doing so. While blockchains aim to enable decentralized systems and applications, our analysis shows that its realization faces challenges in effective decentralization, scalability, naming, and data authentication. These challenges hinder the adoption of blockchain as a foundational solution in metaverse applications. In addition, we perform a comparative study of the Ethereum Naming System (ENS) and DNS.

This paper is organized as follows, with the assumption that readers have a basic knowledge of the metaverse and public-key cryptography but not necessarily with blockchain. Section 2 describes a simple metaverse application of a tea party between two users. Section 3 provides an overview of the blockchain and blockchain-based name mapping systems and identifies potential challenges in the operations of such naming systems at scale. Section 4 discusses the root cause of the identified scalability and cost challenges through a comparison of ENS and DNS, namespace uniqueness, and the role of the blockchain in trust management. Finally, Section 5 concludes the paper.

2. A Simple Metaverse Application: Tea Party

In this section, we use an example application to explain the functional requirements needed to enable secure transactions of objects between different entities in the metaverse.

Alice and Bob meet at a rendezvous point in the metaverse to have a tea party. Alice accepts only green tea from Bob, while Bob accepts either green or black tea from anyone. This simple use case raises two requirements:

1) The tea serving transaction requires the tea object to be transferred from one entity's pot to another's cup.
2) Each entity has the ability to allow or deny the tea from another entity.

The following elements need to be defined to create such a transaction in a secure way [5]:

1) Interacting entities should have *identities* in the metaverse and the ability to *authenticate* each other.
2) Each party should be able to define its own *authorization policies* in securing transactions.
3) All transactions should be recorded to enable *auditing*.

2.1. Naming

We use the following namespace for our application:

/prefix/object/content/[2]

1. There are two types of blockchains: permissioned and permissionless blockchains, referred to in this paper as identity-based and anonymous blockchains. This paper focuses on understanding the capability and limitations of anonymous blockchains.

2. We use '/' notation for readability purposes only. /prefix/object/content is equivalent to content.object.prefix.

(e.g. */alice/tea_pot/black_tea*, or */bob/cup/green_tea*)

/alice/tea_pot/green_tea is a tea object owned by Alice, which is contained in 'tea_pot' and has an associated amount.

The names defined above are *semantically* meaningful: a name both uniquely identifies an entity or object and describes its relationship with other entities in proper context. The semantics in the names are critical in defining the authentication and authorization policies supporting application logic. Today, semantically meaningful names are ubiquitously used across internet applications, such as web and email, to securely and meaningfully identify entities.

2.2. Authentication

In order to support authentication, *identity* must first be established. Every entity needs to obtain a name, then establish its ownership of the name in the form of a certificate from a certificate issuer. The certificate issuer binds the entity's semantically meaningful name to its public key.

Once Alice and Bob obtain their identity and before they can transact in the metaverse, they need to authenticate each other according to their own trust policies. Alice can define her authentication trust policy as shown below, which indicates that any object created by Bob should be authenticated with a specific trust anchor[3].

$$/alice \rightarrow /bob$$

The above certificate chain starts from the trust anchor (left) and ends at the certificate owner (right); '$\rightarrow$' indicates that Alice issues a certificate to Bob.

Alternatively, a commercial certificate authority (CA) may be used as a trust anchor (e.g. Web PKI)[4]. In this case, Bob defines the following authentication policy:

$$/root\_CA\_A \rightarrow /intermediate\_CA\_B \rightarrow /alice$$

There exists one important difference between this trust policy and today's Web PKI: while the latter accepts certificates issued to any name owners by any of the certificate authorities, Bob's security policy determines which trust anchor(s) is valid for which *specific namespace* (e.g. Alice's certificate will be accepted as valid only if the trust anchor is the one Bob designates, in this case /root_CA_A).

In our example, owners sign the exchanged tea objects, enabling recipients to authenticate all received objects according to their own policies. Alice and Bob also define their authorization policies for the tea party interactions after object authentication, as we describe next.

2.3. Authorization

Continuing our example, Alice creates an authorization policy to define *who* can pour *what type* of content to her cup:

/bob/tea_pot/green_tea ***can pour() to*** */alice/cup/**

The pouring event is defined via the *pour(content_name, target_obj, amount)* function, which will transfer a specified amount of content to the target object.

Bob has a more relaxed authorization policy of accepting both types of tea from anyone, so he defines authentication policy as follows:

//tea_pot/green_tea OR black_tea* ***can pour() to*** */bob/cup/**

Based on the his authentication and authorization policies, if Bob receives a tea object from an entity claiming to be Alice with a certificate chain terminating at trust anchor /root_CA_A, he will accept it; but if the trust anchor is something else, e.g. /root_CA_C, then the received tea object cannot be authenticated, thus not accepted.

The transaction specified in the authorization policies will be performed with the explicit permission of the user, as specified in their trust policies, in real-time. However, if a *pour* request does not comply with the policies, it will be automatically rejected.

We emphasize that all authentication and authorization policies are defined by Alice and Bob individually, because the question of *"who trusts whom for what?"* must be answered by each party in order to have secure transactions in a decentralized metaverse environment.

2.4. Transaction Recording

When Alice pours tea into Bob's cup, the transferred tea becomes part of Bob's belongings. This transaction should be recorded, which Bob can use as proof of the transaction any time as needed. Furthermore, transaction logging is critical to prove data provenance [2]. In case a malicious input from Alice could exploit a vulnerability to compromise Bob's system, which could even further spread to other entities that interact with Bob [7], recording all transactions can help identify the entry point of attacks, allowing malicious actors to be identified and quarantined in a timely manner.

3. Blockchain's Role in Metaverse Trust and Transactions

In this section, we first provide a brief background on blockchains, and then introduce Ethereum blockchain and the Ethereum Name Service (ENS); we use Ethereum and ENS an example to articulate the blockchain's capabilities in supporting secure transactions of our metaverse tea party application case. Specifically, we analyze Ethereum from the aspects of data authentication, system scalability, and the truthfulness of its decentralization capability, to reveal open issues in using blockchain-based solutions to support metaverse.

3. A *trust anchor* refers to a self-signed certificate that has been verified out-of-band, thus can be used to verify other certificates.

4. Other possibilities include a single global trust anchor model (e.g. DNSSEC), or PGP [6].

3.1. Background

A blockchain[5] is an immutable ledger that is replicated at every participating node in a network. Its immutability is achieved via hash-chained blocks in which each block contains a cryptographic hash of its data and the hash of the preceding block. This hash-chaining prevents the tampering of on-chain data, because changing any block will lead to either changing all subsequent blocks, or otherwise being detected. Each block contains a list of transaction records stored in a Merkle tree to enable efficient verification of the records' integrity [8].

There are typically two types of participants in a blockchain network: full nodes and simple nodes. Full nodes are responsible for collecting transactions, verifying the transactions, adding new blocks to the chain, and keeping the entire chain history. In contrast, simple nodes perform two functions: sharing with full-nodes the transactions to be added to the blockchain, and querying full nodes to obtain on-chain data, such as fetching a particular block or getting a Merkle tree for transaction verification [9].

Blockchains can be categorized into two types: *identity-based* blockchains and *anonymous* blockchains. In identity-based blockchains, nodes on the network need to have semantically meaningful name as identities (as described in 2.2), so that they can be authenticated and authorized to participate in the blockchain network by its governance entity. In anonymous blockchains, nodes are identified by the hash of their public key[6]. An entity E uses its private key to sign its data, which can be authenticated using E's public key specified in its identity. This form of identity is referred to as a self-certifying name. Since these names cannot be directly used as-is to authenticate a real-world identity, anonymous blockchain cannot perform authorization of participants. This lack of authentication leads to the potential Sybil problem: any single real-world entity can masquerade as multiple entities in a blockchain system. In order to defend against Sybil attacks, instead of using a majority vote, anonymous blockchains use a probabilistic election mechanism as the gating function to approve the addition of each new block. To append a new block to the chain, this mechanism requires candidates to either perform an expensive operation, or to show the possession of large quantities of resources. The two most prominent examples of gating functions are proof-of-work and proof-of-stake.

Proof-of-Work (PoW) requires a computationally operation [10]. Participants compete with each other in running hashing functions in a brute-force manner to find a nonce that results in an appropriate hash. PoW's high energy consumption and slow transaction rate eventually led to other types of gating functions such as Proof-of-Stake.

Proof-of-stake (PoS) relies on staked assets. In each round, every node first stakes some tokens. A node's probability to become the block proposer in a round is proportional to its total amount of staked tokens. Malicious actions, once detected, are penalized by having an amount of the staked tokens removed[7] [12].

Gating functions coordinate the network so that only one node is authorized to add the next block to the chain at a time. However, since they are probabilistic in nature, it is possible for more than one node propose next block in one block proposal period, resulting in the chain forking into multiple branches. This forking problem is mitigated by probabilistic block finality, in which the probability of the separate branches of the same chain growing in parallel decreases exponentially with each consecutive round [10]. Hence a block, and the transactions therein, is only accepted as finalized after a certain number of blocks have accumulated on top of it. In addition, since an anonymous blockchain provides a flat, immutable, and replicated ledger for *the entire system*, the block proposal period must be long enough to allow the newly proposed block to be disseminated to the entire network before next cycle. Compounded with the expensive gating function, anonymous blockchains typically have much lower transaction throughput compared to identity-based blockchains [11]. Furthermore, we also note that the anonymity nature makes the chain unable to verify the contents of blocks being added. Therefore, misbehavers can only be identified by external means.

3.2. Case Study: Ethereum

Ethereum is an anonymous blockchain with compute capability, in which the ledger is used to maintain a state machine among nodes. Ethereum's compute environment, the *Ethereum Virtual Machine* (EVM) is a stack-based virtual machine that executes programs stored on-chain called *smart contracts*. These smart contracts change the state of the virtual machine and support complex transactions between on-chain entities and blockchain-based applications (decentralized applications, or *DApps*). Nodes agree on the order of execution of smart contracts in order to maintain the same EVM state globally [13], [14]. Since many chains extend Ethereum's runtime to be EVM-equivalent or EVM-compatible, Ethereum provides a relevant blockchain implementation to discuss the challenges of a blockchain-supported metaverse.

Supporting the trust policies described in Section 2 requires semantically meaningful names, thus anonymous blockchains need a mapping system to map semantic names to flat, semantic-free blockchain identifiers. In recent years, multiple blockchain-based systems have developed name services to meet this need, such as Ethereum Name Service (ENS) [15], Unstoppable Domains [16], and Handshake [17]. We use ENS to demonstrate a currently implemented mapping service, and examine operational challenges in a widely deployed metaverse.

5. This paper uses the term blockchain to refer to blockchain's implementation as a distributed ledger.

6. This is not to be confused with wallets that identify users of the blockchain. Although they are also identified by their public key hashes, they do not participate in the network infrastructure or block proposal.

7. For example, in voting based PoS algorithms(e.g Casper FFG), nodes stake an amount of token to become a validator for voting for a particular block. In the case of double voting, violator's staked coins are slashed [11]

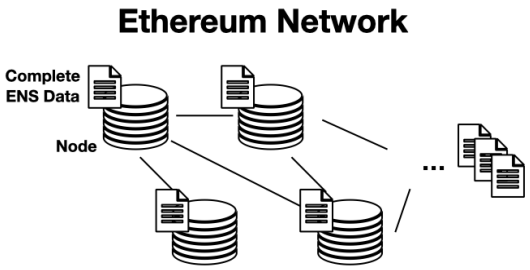

Figure 1. Replication of ENS across nodes in the Ethereum network.

Ethereum Name Service. The Ethereum Name Service is a widely-adopted naming service, with over approximately 2.8 million active names [18]. ENS follows the DNS naming hierarchy, with a root domain at the top, *.eth* as a top-level domain (TLD)[8], and allocate names under the *.eth* domain. An allocated ENS domain name looks like:

mysubdomain.mydomain.eth.

In addition to *.eth*, ENS also has another TLD in use: .test (only for use on an Ethereum test network). The appearance of all other TLDs are imported from DNS namespace [15].

ENS closely follows DNS's name ownership and delegation model. Names in ENS have a *registrant* and a *controller*. A registrant owns the registration of an ENS domain. It can delegate and reclaim its subdomains. The controller of an ENS domain D has the power to edit the records associated with D, and can transfer the control of the name to other entities, but cannot change the registrant or create further subdomains. All ENS names (domains and subdomains) are registered in the *ENS registry* [19] smart contracts (Figure-2).

ENS maps hierarchically structured semantic names to blockchain identifiers and other metadata. ENS names are resolved to its metadata through smart contracts. The two smart contracts that carry out the mapping are a *registry* and a *resolver*. The registry maps names to their respective resolvers. There is only one registry for the ENS namespace: the *ENS Registry*. Resolvers resolve domains and their subdomains to the appropriate metadata. Names are resolved by an ENS public resolver by default, although domain owners can also configure and deploy their own resolvers. Resolvers, like all smart contracts, are identified by their on-chain address (the hash of it's public key). Resolution is not necessarily hierarchical, as names can be resolved by going directly to its resolver from the registry.

Different from DNS, ENS's authentication does not follow the name hierarchy, but is based on the relations recorded in the blocks stored in the chain, and on the immutability of the chain in recording the history of all name registrations. The EVM also must be a trusted execution environment for the smart contracts that register and resolve names. ENS's smart contracts are owned by ENS DAO[9], with the ownership of the ENS root domain held by a multi-signature smart-contract controlled by 7 administrators.

8. *.eth* is not registered with ICANN as a top-level domain.

9. "DAO" refers to a Decentralized Autonomous Organization, a governance body which is not considered a central authority [13].

3.3. Challenges

Although blockchain provides strong guarantees as an immutable distributed ledger, its architecture and limitations provide challenges to becoming a globally ubiquitous infrastructure. In this section, we discuss the following challenges:

1) Data Authentication
2) Scalability
3) Centralization

Data Authentication Querying Ethereum nodes (and by extension, ENS) is not free. In order to make a read-only call that does not alter the state of the EVM (such as the resolution of a name), developers can either run their own node, which can be expensive, use a node-as-a-service platform to provide a node to service calls to the network, or use a paid 3rd-party query service. To authenticate the query results obtained from third-party services providers, callers must operate their own *simple-node* (as defined in section 3.1) or higher in order to verify the block-headers and cross-reference state of the chain with the result given. Therefore, authenticating on-chain data is not a trivial task.

Scalability Every full node on the Ethereum network is responsible for running a complete instance of the Ethereum distributed state machine. These nodes must replicate all activity on the network in order to maintain a synchronized state machine. With a current network size of 11,000 nodes, all on-chain code and transactions are replicated 11,000 times (Figure-1) [20]. The adoption of Ethereum and the use of DApps add pressure to every individual node and further raise the threshold for qualification of a full node.

An increase in network demand also raises the threshold for users to interact with the chain. As network demand increases, the fee required to execute a transaction (or make a record on the chain) increases. An unreasonable increase in cost, or the inability of a chain to scale and keep fees low, would prevents users from interacting with the chain. An inability to keep latency low will prevent users from experiencing the metaverse in real-time. The blockchains that support the metaverse will have to maintain low fees and latency at a massive scale.

To address the scalability issue, *layer 2* blockchains have been developed to increase transaction speed, lower transaction cost, or a mix of the two. Layer 2 blockchains execute and record transactions on their own chain and store digests of these transactions in their *layer 1* chain, in this case Ethereum. Layer 2 blockchains have independent security and trust assumptions from the layer 1 chain, and so trust in both the layer 2's security and the layer 1's security are a prerequisite to their use. In addition to the improved performance compared to the layer 1, layer 2's also alleviate demand from the layer 1 to keep its latency and cost down.

Query services also exist to scale the data availability of a blockchain. Query services aggregate and serve blockchain data to consumers, removing the need for a node to query

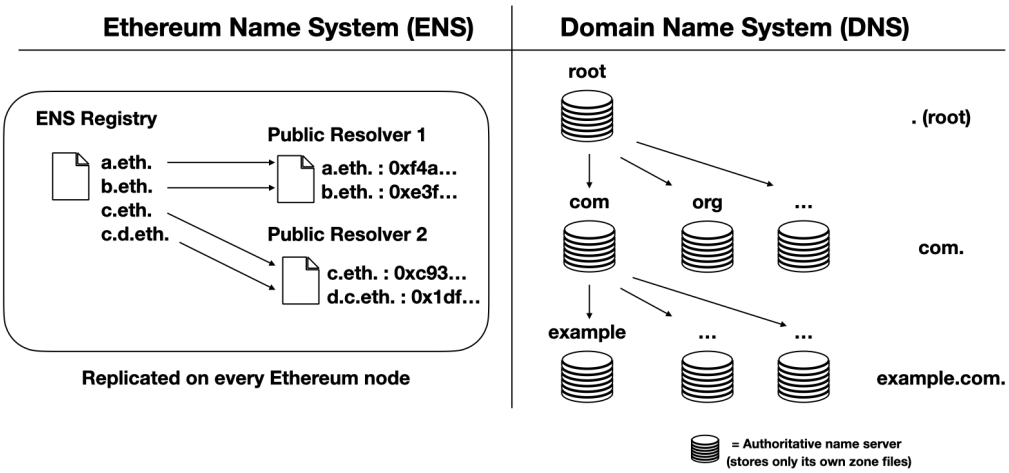

Figure 2. Comparison of ENS smart contract architecture with DNS distributed architecture.

on-chain data. Since query services do not need to store the data they aggregate in a Merkle tree, queries of can be done more efficiently, especially in the case of historical data. The use of data from these services imply trust in both the chain and in the query service.

Centralization. Anonymous blockchains are designed with the goal of achieving decentralization among nodes in the network. However, gating functions are biased towards the favoring most powerful entities in the network, creating a centralization risk [21]. For example, Ethereum's proof-of-stake favors participants with more coin staking capacity while Bitcoin's proof-of-work favors participants with more hashing power. As a result, powerful actors can gain more block rewards and transaction fees than the others and increase their capacity to dominate the network. Small actors can choose to operate as a part of a larger pool to optimize returns, further contributing to centralization. Today, this trend is observable in most popular blockchains. For example, more than 50% of the Ethereum validator capacity is controlled by 8 staking pools [22], while more than 50% of the hashing rate of Bitcoin is controlled by three mining pools [23]. Mitigating centralization risk is crucial in a blockchain ecosystem since whoever controls 51% percent of the network can overwrite previous blocks, thus nullifying the immutability guarantee given by the blockchain (e.g. double-spending attack) [24].

4. Discussion

In this section, we identify the challenges in operating a blockchain system to serve metaverse. We articulate the root cause of the identified scalability and cost challenges in ENS through a comparison with DNS and examine the role of an immutable ledger (i.e. a blockchain) in trust management.

4.1. ENS-DNS Comparison

From the description of ENS in Section 3.2, one can identify several fundamental differences between ENS and DNS (Figure-2).

First, DNS name registration and lookup are completely distributed. Each domain owner is the sole authority for name assignment in its domain, and each domain owner runs their own DNS nameserver(s) to provide lookup services only for its assigned names [25]. In contrast, all allocated ENS names are stored in the blockchain, which is replicated on every node in the network.

Second, the DNS namespace is a shallow tree. Looking up a DNS name is a top-down search starting from a root domain name server, and the search walks down one branch of the tree just a few hops to reach the server with the answer. In contrast, performing an ENS resolution does not involve a traversal of a hierarchical tree but a flat, two-step lookup: first querying the ENS Registry to obtain a name's resolver, and then querying the resolver.

Running a DNS server is cheaper compared to running an Ethereum node, since an DNS nameserver can dedicate its hardware to the singular task of name resolution. A node provisioned with the intent of ENS lookup will need support the computational overhead of being a node, such as continuously querying the network to stay up-to-date with all on-chain activity. Furthermore, a developer intending provide ENS lookup in their application will need to ensure that their node is reachable from all deployments of the application, which will require DNS.

Third, DNS domain owners are incentivized by necessity to deploy their name servers to make their name reachable to users, and their cost is low. In contrast, ENS name and domain owners are not responsible for the infrastructure supporting their name(s). Each full[10] Ethereum node hosts

10. Ethereum has a further category of an archival node, which stores the entire history of the chain. Ethereum full nodes keep the most recent 128 blocks.

not only the entire namespace but also all history of changes, and each name resolution requires the use of the Ethereum network's resources. Nodes operators are incentivized by transaction fees and block rewards to record and serve blockchain-related data, e.g. ENS name registration and resolution.

Despite the above differences, ENS and DNS also share an important commonality: both systems assure name uniqueness by having a governing body control the root domain of the namespace. The Internet Corporation for Assigned Names and Numbers (ICANN), a global non-profit organization, governs DNS. It controls the assignment of unique top level domain names under the root domain. ENS DAO governs ENS. Governance decisions are made via a voting system among holders of the DAO's governance token. ENS's root domain is controlled by a multisig contract whose keys are held by seven people.

4.2. Namespace Uniqueness

Running metaverse applications across entities from different domains requires that all entities have unique names. In today's internet, all applications use of uniquely assigned DNS names. However, today's different blockchain naming systems (e.g. ENS, Unstoppable Domains) operate independently from each other, have top-level domain names that are not registered with ICANN, and do not have an overarching body to coordinate their different semantic namespaces. Unstoppable Domains has offered several TLDs such as .bitcoin, .wallet, and .blockchain in its own namespace. ENS, on the other hand, limits ENS-native name registration to be all under .eth, and supports the import of DNSSEC-enabled DNS names.

Even if one could assume that the names from different blockchain-based naming systems will continue to be unique, having multiple name resolution systems can also complicate application development, since applications need to understand which naming system should be used for which given blockchain-based domain. If we assume that all blockchain systems will follow the practice of ENS (allocating names under a single TLD only), and that eventually their specific TLD name will have been registered with ICANN, then a blockchain-based metaverse application could start its name resolution with the DNS root and convert to a blockchain based naming system when it hits a blockchain TLD (e.g. .eth). This would allow blockchain-based semantic names to get resolved by off-chain entities.

4.3. Blockchain's Role In Trust Management

Anonymous blockchains provide a trustless platform for untrusted parties to maintain a replicated ledger state. In spite of having untrusted actors, on-chain transactions still require authentication to occur. In our example, an entity, /alice, wants to deliver digital tea to /bob. In order to make this on-chain transaction, Alice first needs to know how to address /bob, which requires Bob's semantically meaningful name to be mapped to his public key binding and be endorsed by someone trusted by Alice. In the case of ENS, entities such as Alice trust the ENS smart contracts that define the name-key bindings and the infrastructure that serves those bindings. Alternatively, metaverse applications might adopt a decentralized trust model in which name-key bindings can be defined in different forms (e.g. in a web-of-trust fashion) by different parties and then store these name-key bindings on the blockchain for immutability. In both the case of ENS and decentralized trust management, trust relies on the entity that created the binding, not on the blockchain itself.

5. Conclusion

Anonymous blockchains have been viewed as opening a new direction to build decentralized and secure systems. In this paper, we verify this commonly accepted view through a set of basic functional requirements needed in a simple metaverse use case, which illustrates the necessity of using semantically meaningful names in securing transactions in a decentralized metaverse. We then analyze how well blockchain-based solutions may provide those functions.

Using Ethereum and ENS as a representative example of blockchain-based systems, we show that, although ENS developed a solution for mapping semantic names to blockchain's self-certifying names, the solution records name allocations only; these immutable recordings provide a highly replicated and publicly accessible logging of all name allocations. The logging enables, but is not equivalent to, authentication and authorization by itself for the following reasons: i) Trust relationships are defined *externally*, and then stored on-chain for immutable recording; ii) the mapped content is not validated according to the polices to be used in performing authentication and authorization.

We also show that blockchain-based solutions face scaling challenges which stem from the blockchain's design of recording and replicating the complete history in all full nodes. Our analysis further shows that blockchain-based solutions may require centralized functions in name lookup, lacks assurance for name uniqueness and accessible data authentication at least for the time being.

Although blockchains have been viewed as an enabler for decentralized applications and considered as providing security solutions for decentralized metaverse, the set of open issues revealed by this work raises a question mark to the current capabilities that blockchains provide. As next step, we plan to clearly identify the *root cause* of the identified open issues with blockchain-based security solutions, hence to further deepen our understanding on both the capabilities and intrinsic limitations of blockchain-based systems.

References

[1] "The Metaverse is the Future of Digital Connection | Meta." [Online]. Available: https://about.meta.com/metaverse/

[2] "Metaverse landscape & outlook series." [Online]. Available: https://versemaker.org

[3] A. Al-Ghaili, H. Kasim, N. M. Al-Hada, Z. Hassan, M. Othman, T. Jakir Hussain, R. Kasmani, and I. Shayea, "A Review of Metaverse's Definitions, Architecture, Applications, Challenges, Issues, Solutions, and Future Trends," *IEEE Access*, vol. 10, pp. 125 835–125 866, Dec. 2022.

[4] "Lamina1 white paper." [Online]. Available: https://www.lamina1.com

[5] B. Lampson, "Computer Security in the Real World," *Annual Computer Security Applications Conference*, vol. 16, Dec. 2000.

[6] S. Garfinkel, *PGP: Pretty Good Privacy*. "O'Reilly Media, Inc.", 1995, google-Books-ID: cSe_0OnZqjAC.

[7] Z. Li, Q. A. Chen, R. Yang, and Y. Chen, "Threat Detection and Investigation with System-level Provenance Graphs: A Survey," Dec. 2020, arXiv:2006.01722 [cs]. [Online]. Available: http://arxiv.org/abs/2006.01722

[8] R. C. Merkle, "A Digital Signature Based on a Conventional Encryption Function," in *Advances in Cryptology — CRYPTO '87*, ser. Lecture Notes in Computer Science, C. Pomerance, Ed. Berlin, Heidelberg: Springer, 1988, pp. 369–378.

[9] A. M. Antonopoulos, *Mastering Bitcoin: unlocking digital cryptocurrencies*. " O'Reilly Media, Inc.", 2014.

[10] S. Nakamoto, "Bitcoin: A Peer-to-Peer Electronic Cash System."

[11] Y. Xiao, N. Zhang, W. Lou, and Y. T. Hou, "A Survey of Distributed Consensus Protocols for Blockchain Networks," *IEEE Communications Surveys & Tutorials*, vol. 22, no. 2, pp. 1432–1465, 2020, arXiv:1904.04098 [cs]. [Online]. Available: http://arxiv.org/abs/1904.04098

[12] "Proof-of-stake (PoS)." [Online]. Available: https://ethereum.org

[13] A. Antonopoulos, *Mastering Ethereum*. O'Reilly Media, Inc., Mar. 2023, original-date: 2016-08-10T15:07:54Z. [Online]. Available: https://github.com/ethereumbook/ethereumbook

[14] "Ethereum development documentation." [Online]. Available: https://ethereum.org

[15] "ENS Documentation." [Online]. Available: https://docs.ens.domains/

[16] "Unstoppable Domains." [Online]. Available: https://unstoppabledomains.com/

[17] "Handshake Developer Documentation." [Online]. Available: https://hsd-dev.org/

[18] M. Inoue, "ENS." [Online]. Available: https://dune.com/makoto/ens

[19] N. Johnson, "ERC-137: Ethereum Domain Name Service - Specification." [Online]. Available: https://eips.ethereum.org/EIPS/eip-137

[20] etherscan.io, "Ethereum Node Tracker | Etherscan." [Online]. Available: http://etherscan.io/nodetracker

[21] A. E. Gencer, S. Basu, I. Eyal, R. van Renesse, and E. G. Sirer, "Decentralization in Bitcoin and Ethereum Networks," Mar. 2018, arXiv:1801.03998 [cs]. [Online]. Available: http://arxiv.org/abs/1801.03998

[22] "Staking Pools Services Overview - Open Source Ethereum Blockchain Explorer - beaconcha.in - 2023." [Online]. Available: https://beaconcha.in/pools

[23] "Pool Stats - BTC.com." [Online]. Available: https://btc.com/stats/pool

[24] "51% Attacks," Jul. 2020. [Online]. Available: https://dci.mit.edu/51-attacks

[25] P. Mockapetris, "RFC 1034 DOMAIN NAMES - CONCEPTS AND FACILITIES." [Online]. Available: https://www.ietf.org/rfc/rfc1034.txt