Automata: Privacy-First and Cross-Chain Compute Protocol

Automata Team

Abstract

The booming of Internet-based businesses has brought us to a juncture where important questions have arisen on where we are headed. While enjoying the great benefits from instant inter-connectivity, highly customized services, as well as "free of charge" offerings, the darker side of the evolution also manifests. Lack of transparency in data ownership, dictating censorship criteria, relentless intrusion into personal privacy, just to name a few. All of these are calling for a fundamental paradigm shift in terms of how businesses, users, and data shall interact with each other.

As blockchain technology starts to gain both attention and traction, the idea of a decentralized Web, or Web 3.0, sounds particularly promising. It seems to promise a future where users collectively own and govern applications and businesses that they interact with. When that vision is realized, we will no longer worry about data and privacy matters, as no one would be setting the rules other than the users themselves. The proliferation of decentralized finance (DeFi) applications in the past few months seems to have accelerated the transition into Web 3.0.

Nevertheless, reality checks would state otherwise. Up to now, there is a fundamental divide between what is available for Web 3.0 and what is needed to build the Internet-grade applications today. For one, the transparency of blockchains albeit pushing for greater accountability in decentralized applications, can easily be abused to violate user privacy. A typical example is all these frontrunning tricks that will cause victim users of losses, large or small, when they are trading with decentralized exchanges today. Moreover, the intrinsic nature of redundant computation in blockchains could become prohibitively costly when applied in scale. Imagine high performance e-commerce applications that need to process tens of thousands of transactions per second, and they would not be able to wait for several blockchain nodes to each "double check" the results and try to agree with each other.

To solves these roadblocks to make Web 3.0 practical and usable at the Internet-grade, we develop Automata, a high performance compute protocol that empowers Web 3.0 applications and businesses with privacy-first, high assurance and friction-less transactions. Backward-compatibility with today's existing decentralized applications is a key design criterion. The mission of Automata is to provide the necessary primitives, functionalities as well as infrastructures to make realizing the Web 3.0 vision a reality. In this paper, we outline how we plan to achieve

this mission via the Automata system.

1 Introduction

With the presence of the Internet dominance of IT giants, like Google [1], Facebook [2] and Amazon [3], users get used to providing their personal data to these companies with or without their consent, in exchange for convenient services like customized recommendations of news feeds and online shopping products. The recent Facebook–Cambridge Analytica data scandal drastically attracts users' attention to their privacy, as Facebook let Cambridge Analytica harvest the personal data of millions of people's Facebook profiles without their consent [4, 5, 6]. Users may have trust in Facebook, but they would not have any trust in any unknown third party. While more and more incidents show the evidence that centralized parties can leak users' private data to others, users start to seek alternatives to meet their needs, e.g., blockchains and decentralized applications. Users use blockchains, like Ethereum [7], to transfer their digital assets for payments. They use P2P file-sharing systems like StorJ [8] and IPFS [9], to host their own data, and use P2P microblogging service like Peepeth, to tweet their messages in a decentralized manner. People are willing to take control of their data and privacy. However, trusting unknown nodes in blockchain or decentralized application platforms will not be safer than trusting the centralized and well-know entities, as the execution and storage on the nodes typically are public. In the permissionless blockchain systems, if malicious nodes host the execution or data for the users, they can reveal the private information of particular users.

The impact of privacy leakage in the blockchain space starts becoming severer than what we have thought with the emergence of decentralized finance applications, especially on Ethereum. For the past several years, DeFi applications are the most attractive and successful applications on blockchain platforms, including DEXes like Uniswap [10] and Curve [11] as well as lending protocols like Maker DAO [12] and AAVE [13]. The total value (of digital assets) locked of these applications has surpassed 10 billion USD on Ethereum [14]. However, due to the transparency and lack of privacy protections, the blockchain platforms like Ethereum are becoming the dark forest, where bots and miners can behave as apex predators to frontrun the transactions from ordinary users with higher gas fees and other methods [15]. When finding an opportunity to arbitrage, e.g., buying a digital asset at a low price on a dex and selling at a high price on another dex, the bots or miners can replicate the transactions from users with higher gas fees and their own addresses to let the network prioritize and accept the replicated transaction instead of the first proposed one. Such front-running attacks have occurred in the real world multiple times, and drastically disrupt the execution of on-chain bidding process for numerous digital assets [16], which completely bias the fairness of these decentralized systems and cause huge financial damage to blockchain users.

Newly launched projects have tried to apply various privacy-related primitives to blockchain systems. For instance, Zcash [17] leverages Zero-Knowledge Succinct Non-Interactive Argument of Knowledge (zk-SNARK) [18, 19, 20, 21] to provide anonymity for senders and receivers. Monero applies ring signature to achieve the similar anonymity. For more general-purpose decentralized applications, Enigma [22] and Oasis [23] use trusted hardware say Intel SGX to run private smart contracts in a trusted enclave. This can achieve better performance compared to zero-knowledge proof systems [24] and secure multiparty computation [25]. However, these approaches can only provide *confidentiality*, for which the data itself is hidden against malicious nodes. They do not ensure a high degree of *privacy*, which prevents malicious nodes from inferring users' private information (such as secret keys and user queries [26, 27]) based on the data access patterns, like the frequency of accessing a particular region of RAM/disk or a storage node in a P2P system. Since these platforms are permissionless, and malicious nodes can easily join the system to execute contracts or host data, these nodes can further launch attacks to infer users' personal information, which is against the expectation of users that decentralization is a safer place for privacy than a centralized entity. Privacy in decentralized applications is a zero-sum game. If the decentralized platform cannot provide better privacy, why can users have more trust in unfamiliar nodes in the network than the properly registered centralized entity?

In this work, we use a combination of our previous work *OblivP2P* [28], *PRO-ORAM* [29] and *SGXP2P* [30], and base on state-of-the-art algorithms (e.g., Oblivious RAM) and trusted hardware (e.g., Intel SGX), to protect user privacy by concealing data access patterns on a single node as well as multiple nodes in a P2P system. To further enrich the blockchain ecosystem, we propose Automata, a privacy-first cross-chain compute protocol for decentralized applications. The strong privacy can guarantee that the remote nodes cannot leak any information such as which data is being accessed, when the data was last accessed and the access pattern is sequential or random, etc.

In Automata, capable machines can join the protocol as two type of nodes, Automata Validator and Automata Geode, contributing to decentralization, security and performance of the protocol.

Automata Validator is required to stake a decent amount of tokens and in charge of various control-plane tasks, e.g., registration of Automata Node and Automata Geode from community, attestation of Geode hardware and software, marketplace for selling and buying computing power of Geode, fulfilment of compute tasks requested by service vendors, as well as rewarding Geode that complete the tasks.

Automata Geode provides a shielded, unbiased and decluttered computation environment, to execute tasks scheduled by the control plane. Geode utilizes the state-of-the-art hardware-based trust along with algorithm-based hardening to preserve a high degree of privacy, not only protecting the data directly but also making the data access patterns oblivious to Geode provider. Geode is able to connect to different ledgers or blockchains to co-operate the same piece of data or transfer data across chains in a privacy-preserving manner.

We envision AUTOMATA to provide high applicability for decentralized applications to achieve versatile functionalities apart from the present token-related activities.

Contributions. We summarize our contributions below:

- **Privacy-first cross-chain service plane.** AUTOMATA empowers decentralized applications with privacy services connecting various blockchains. Service vendors can build tooling services using Geode or integrate Geode primitive into applications to let users take control over their own data.
- **Shielded and unbiased compute plane.** AUTOMATA enables high degree of privacy for computation in Geode, which conceals data access patterns against the hosting nodes themselves on a single machine as well as in a P2P system. The provided privacy can be even stronger than centralized cloud services, e.g., AWS and Google cloud platform.
- **Elastic and scalable control plane.** AUTOMATA provides an elastic control plane operated by a group of staking nodes, which safeguards the entire protocol and governs the interactions between various participants in the protocol.
- **Collaborative reward mechanism for staking and hosting.** AUTOMATA provides a pay-as-you-go service for service vendors seeking privacy-first compute resources. Geode providers will earn reward from staking as well as hosting the Geode. AUTOMATA validators and other roles can earn staking rewards as compensation for their cooperation and maintenance cost.
- **Unified cross-chain support for interoperability.** AUTOMATA provides interfaces for multiple blockchains to exchange data and cooperate on the agreed set of data in an atomic and privacy-preserving manner.
- **Verifiable accounting mechanism.** We use various accounting methods to calculate the reward of Geode when it completes the assigned tasks. The payment method can be a pay-as-you-go model, which allows users to pay for a computation per usage, or timely manner like hourly, daily, monthly or yearly.
- **Applicability and Speed for decentralized applications.** The ultimate goal of AUTOMATA is to provide toolings for developers to build powerful applications without explicit constraints, which can compete with web or mobile applications in general.

2 Overview

AUTOMATA has four planes, i.e., ledger, control, compute, and service plane, as shown in Figure 1. The ledger plane represents various blockchain systems,

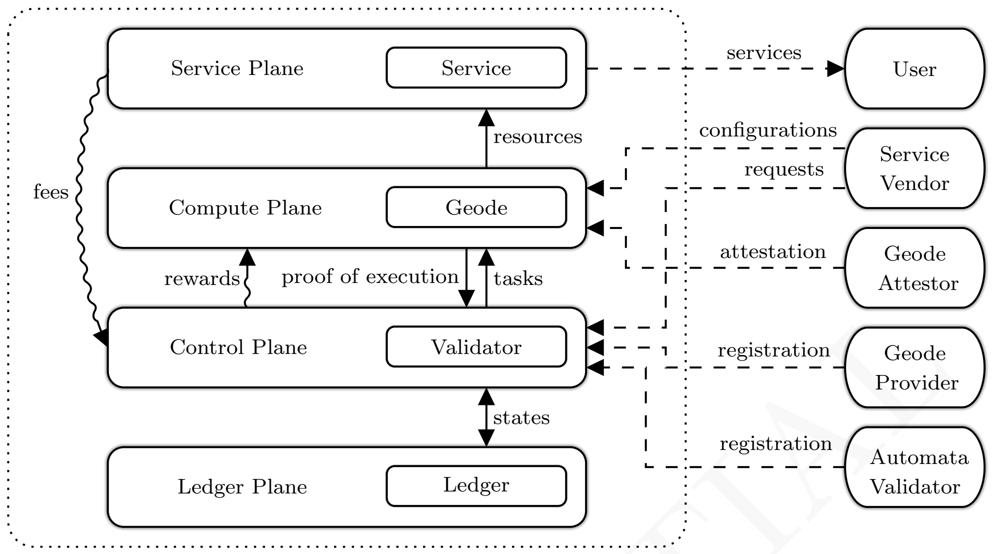

Figure 1: The overview of AUTOMATA architecture

e.g., Bitcoin and Ethereum, which can act as a global distributed ledger system for the purpose of payment settlement and task registration. The control plane is the coordinator to outsource tasks, distribute rewards (like gas fees) from ledger plane to compute plane, manages the registration of AUTOMATA Validator and AUTOMATA Geode with stakes, and evaluates the performance of these nodes. The control plane consists of a group of validator to ensure the security and integrity of the system. The compute plane is for Geodes to accept and accomplish tasks assigned by the control plane. Any machine that meets the requirements of trusted hardware such as SGX [31] or TrustZone [32], can join AUTOMATA to become a Geode.

One of the main goals of AUTOMATA is to ensure the privacy of AUTOMATA Geode. We leverage our previous works *OblivP2P* [28], *PRO-ORAM* [29] and *SGXP2P* [30], as well as the state-of-the-art ORAM and SGX techniques to achieve the obliviousness of the data access patterns for Geodes. The strong privacy can ensure the Geode itself learn no information about 1) which data is being accessed; 2) when the data was last accessed; 3) whether the same data is being accessed (linkability); 4) the access pattern is sequential, random, etc; or 5) whether the access is a read or a write, as described in Definition 2.1.

2.1 Threat Model

In our threat model, we consider a global adversary, which can actively monitor the traces of users to do further analysis. For example, there exist tools like Global BitTorrent Monitor [33] or BitStalker [34] that support accurate and efficient monitoring of BitTorrent. Previous research has shown that any BitTorrent user can be logged within a span of 3 hours, revealing the digital identity and the content downloaded [35]. Meanwhile, some of the nodes in the network can also be controlled by the adversary. They can further collude to exchange data with other adversarial peers in the system, such as observed / served requests and the contents stored at their local storage. In real-world blockchain platforms like Ethereum, the adversary can monitor on-chain transactions using bots or running nodes to frontrun the existing transactions with higher gas. In AUTOMATA, compute plane nodes are required to have trusted hardwares, which guarantee the untampered execution even in the presence of a malicious provider who has privileged control of the underlying infrastructure, such as OS, memory and disks. The adversarial nodes can take arbitrary actions as long as it does not violate the guarantees of trusted hardware (e.g., Intel SGX).

2.2 Background

Square-Root ORAM. Oblivious RAM, introduced by Goldreich and Ostrovsky [36], is a cryptographic primitive that prevents an adversary from inferring any information via the memory access pattern.

The square-root ORAM scheme [36], uses $N + \sqrt{N}$ permuted memory and a $\sqrt{N}$ *stash* memory, both of them are stored encrypted on the untrusted cloud storage. The permuted memory contains N real blocks and $\sqrt{N}$ dummy blocks arranged according to a pseudo-random permutation π. To access a block, the protocol first scans the entire stash deterministically for the block. If the requested block is found in the stash then the protocol makes a fake access to a dummy block in the permuted memory. Otherwise, it accesses the real block from the permuted memory. The accessed block is then written to the stash by re-encrypting the entire $\sqrt{N}$ stash memory. The key trick here is that all accesses exhibit a deterministic access order to the adversarial server, namely: a deterministic scan of the stash elements, followed by an access to a real or dummy block in permuted memory, followed by a final re-encrypted write and update to the stash. After every $\sqrt{N}$ requests, the protocol updates the permuted memory with the stash values and obliviously permutes (shuffles) it randomly. This shuffling step incurs $O(N \log^2 N)$ overhead, resulting in an amortized latency of $O(\sqrt{N} \log^2 N)$ per request.

Tree-based ORAM. Tree-based ORAM introduced by Shi et al. [37] offers a poly-logarithmic overhead which is further reduced due to improvements suggested in the follow up works [38, 39, 40, 41, 42, 43]. In particular, Ring ORAM, [39] is one of the latest improvements for tree-based ORAM. In Ring ORAM, to store N data blocks, the memory is organized in a (roughly) $\log N$-height full binary tree, where each node contains z real blocks and s dummy blocks. Whenever a block is accessed in the tree, it is associated to a new randomly selected leaf identifier called, `tag`. The client stores this association in a position map PosMap along with a private storage (stash). To read and write to the untrusted memory, the client performs an Access followed by an Evict operation described *at a high level* as follows:

- Access(adr): Given address adr, the client fetches the leaf identifier tag from PosMap. Given tag, the client downloads one block per every node in the path $\mathcal{P}(\mathsf{tag})$ that starts from the root and ends with the leaf tag. The client decrypts the retrieved blocks, and retrieves the desired block. This block is appended to the stash.
- Evict(A, ν): After A accesses, the client selects a path $\mathcal{P}(\nu)$ based on a deterministic reverse lexicographic order, downloads the path, decrypts it and appends it to the stash. The client runs the least common ancestor algorithm to sort the blocks as in [38]. Finally, the client freshly encrypts the blocks and writes them back to the nodes in the path.

The stash is upper bounded by $O(\log N)$. The overall bandwidth may reach $\simeq 2.5 \log N$, for N blocks stored. In Ring ORAM, eviction happens periodically after a controllable parameter $A = 2z$ accesses where z is the number of blocks in each bucket [39].

ORAM Security definition. We use the standard security definition for ORAMs [44, 45, 39]. Intuitively, the security definition requires that the server learns nothing about the access pattern. In other words, no information should be leaked about: 1) which data is being accessed; 2) when the data was last accessed; 3) whether the same data is being accessed (linkability); 4) the access pattern is sequential, random, etc; or 5) whether the access is a read or a write.

Definition 2.1
Let $\overrightarrow{y} := ((op_1, u_1, data_1), (op_2, u_2, data_2), ..., (op_L, u_L, data_L))$ denote a data request sequence of length L, where each op_i denotes a $read(u_i)$ or a $write(u_i, data_i)$ operation. Specifically, u_i denotes the identifier of the block being read or written, and $data_i$ denotes the data being written. Let $A(\overrightarrow{y})$ denote the sequence of accesses to the storage given the sequence of data requests $\overrightarrow{y}$. An ORAM construction is said to be secure if for any two data request sequences $\overrightarrow{y}$ and $\overrightarrow{z}$ of the same length, their access patterns $A(\overrightarrow{y})$ and $A(\overrightarrow{z})$ are computationally indistinguishable by anyone but the client.

Intel SGX. Recently, Intel proposed support for a trusted hardware primitive called Software Guard Extensions (SGX). With SGX, we can create isolated memory regions called *enclaves* which are inaccessible to the underlying operating system or any other application. SGX allows the creation of hardware-isolated private memory region or enclaved memory. For SGX CPUs, BIOS allocates a certain region for processor reserved memory (PRM) at the time of boot up. The underlying CPU reserves a part of this PRM to create enclaves. All the code and data in the enclaved memory is inaccessible even to the

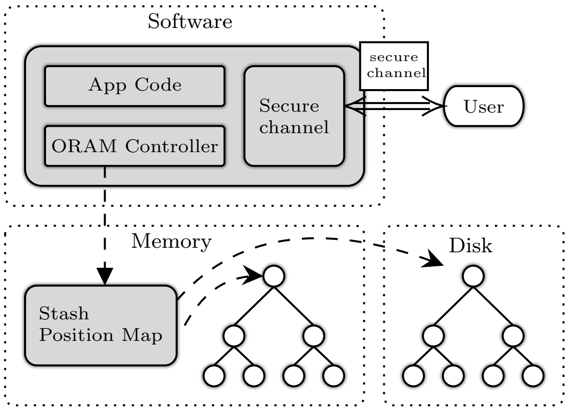

Figure 2: Obliviousness on a single machine.

privileged software such as the OS. It guarantees confidentiality of the private data within enclaves from the adversary. Along with enclaved execution, SGX-enabled CPUs support remote attestation of the software executing within an enclave. This security features enables a remote party to verify the integrity of the software executing on an untrusted platform such as the compute plane. Further, it supports local attestation between two enclaves executing on the same machine. These enclaves can then establish a secure channel and communicate with each other. One can perform such attestation of an enclave program as described in the SGX Programming Reference [31].

3 Automata in Depth

3.1 Privacy-First Cross-Chain Service Plane

Service Plane. The service plane provides privacy-preserving services directly facing the user base from different blockchain networks. Anyone can be a service vendor developing these services targeting users from one or multiple networks, by utilizing the privacy primitives of Geode as well as the ability of running code in native or various virtual machines (e.g., EVM, WASM). With less constraints in Geode, services can do more than then present blockchains, such as running a trusted gateway to forward user inputs to other blockchain services with privacy enhanced, creating a API service to provision information derived from secrets, or even integrating the application closely with Geode for better security and speed.

3.2 Shield and Unbiased Compute Plane

Compute Plane. As shown in Figure 1, the compute plane accepts tasks scheduled by the control plane and submit proof of execution and performance

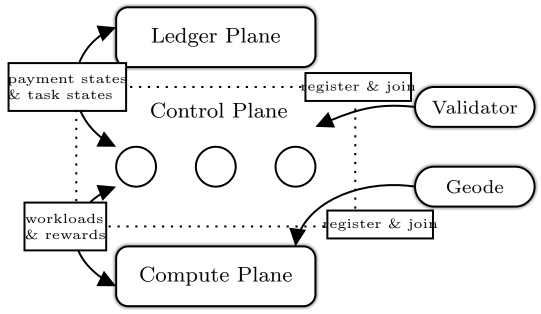

Figure 3: Elastic and scalable control plane

to it to own the reward. We leverage *PRO-ORAM*, *OblivP2P* and other latest ORAM techniques to ensure the obliviousness for data access patterns of the computation and storage in AUTOMATA Geode. More specifically, for any Geode, we use ORAM primitives including *PRO-ORAM* [29] and *ZeroTrace* [46] to hide the access patterns against the hosting machine. The guaranteed degree of privacy is even higher than the one provided by centralized cloud services, e.g., Amazon Web Services(AWS) [47] and Google cloud platform [48], as these platforms still may have the implicit access to users' code and data, or at least they have the privilege to monitor the data access patterns via hypervisor or hardware. Further, with the support of trusted hardware, we can also provide a confidential execution environment for users to run their own algorithms without giving away the license or making code public. This could attract more Web and mobile developers to build their applications in a decentralized and privacy-preserving manner. As shown in Figure 2, via a secure channel, a user can execute code in an enclave and our ORAM controller will manage the data in RAM or disk oblivious against the node itself. Both the user's code and data are protected by our system. According to previous research studies, this approach may introduce 2 - 3x overhead compared to the program executing in an unprotected environment. Considering the benefits of privacy and confidentiality as well as the slowness of the present blockchains (like 5-mins to 1-hour confirmation time), this overhead is acceptable for the vast majority of blockchain users. Meanwhile, AUTOMATA also allows the user to select the desired degree of privacy. If privacy is not a concern, the compute plane can provide a pure confidential execution environment to only ensure the code and data integrity. Based on the gas fees paid by a user and also privacy degree, the control plane can instruct the compute plane to select one or multiple nodes to serve the user for a period of time, e.g., one hour, day and month.

3.3 Elastic and Scalable Control Plane

Control Plane. The control plane is the coordinator between the ledger plane and the compute plane as shown in Figure 3. In the ledger plane, there can be multiple blockchains acting as the payment and task assignment channels for Automata. The control plane receives the tasks and gas fees from service vendors and distribute them to the compute plane. The control plane handles the registration of Automata Node and Automata Geode. For each type of node, the requirement varies, e.g., control nodes are required to have more stakes, compute nodes need powerful trusted hardware.If there's any misbehavior on a control node, the node will be punished by slashing a portion of its stakes. If the node is offline for a long time, it will be removed from the plane and be penalized with a large amount of stakes. To reach consensus on various decisions, e.g., task assignment and penalty for nodes, the control nodes runs Proof-of-Stake protocol safeguard the network. In the future, they can also can run BFT-like protocols or the highly-efficient consensus protocols proposed in *SGXP2P* [30]. Upon the completion of the tasks, the control plane will account the performance of nodes and reward them with the gas fees charged from service vendors.

3.4 Collaborative Reward Mechanism for Staking and Gas

To join Automata as an Automata Node or Automata Geode, an amount of stakes is required to constrain the node from behaving maliciously or being offline without completing tasks. When a user pays for the compute/storage resources as gas fees in the ledger plane, a small portion (e.g., 15%) of gas fees will be distributed to the control nodes to cover their maintenance cost and reward them. The vast majority of the fees (e.g., 85%) goes to Geodes, which provide the resources and services for the user. With more users and developers use the services provided by Automata, both Automata Node and Automata Geode can earn rewards collaboratively.

3.5 Unified Cross-Chain Support for Interoperability

The cross-chain communication with data exchange has been a critical hurdle against the smart contract development for a long time. The primary difficulty lies in the inconsistency of data and block formats for different blockchains, thus it's quite hard for various platforms to support a unified interface to perform cross-chain operations for smart contracts. This service enables contracts not only to exchange data across chains, but also to co-operate on computation and storage in a privacy-preserving manner. This further provides opportunities for developers to build collaborative decentralized applications like Google Docs [49] and Slack [50]. By placing our cross-chain service contracts on multiple blockchains, we can provide a unified interface for contracts to interoperate on various ledgers.

To reward Automata Node and Automata Geode in a more convincing way, we need a better evaluation system for their performance. For the nodes, they can run consensus protocols periodically to evaluate each other's performance such as online time and # of signatures. For the Geode in compute plane, we apply the various techniques [51], especially using the primitives provided by Intel SGX [30] to quantify the resources used by the compute node, e.g., CPU time, memory, I/O bytes and network bandwidth. For the Geode with storage, apart from the above methods, we can also leverage proof of retrievability [52, 53] to verify the availability of the data. To ensure data availability, the on-demand replica mechanism can provide multiple replicas for users to meet their requirements. Further, advanced algorithms for data availability or recovery may also be applied to achieve more efficient data fetch/store in this decentralized system [54, 55, 56].

Automata provides privacy-first compute resources for service vendors to develop full-fledged and versatile Dapps, which can compete with web/mobile apps. Developers will not face the constraints in present blockchains, such as limited set of features or APIs, long confirmation time. When a developer creates a website-like Dapp on Automata he or she can perform the payment on any supported ledger and the compute resources will be used to serve users who view the site. When web users click the advertisements on the site, the developer is encouraged to share a small portion of the revenue with the control plane of Automata in return, to further support the system maintenance. Besides website, developers can also build privacy-preserving file sharing Dapps, and various decentralized finance apps like DEX to achieve low-latency and prompt user experience and compete with centralized exchanges.

4 Related Work

	Decentralized Compute	Confidential Compute	Privacy-preserving Compute	Decentralized Storage	Confidential Storage	Privacy-preserving Storage
IPFS	✗	✗	✗	✓	✓	✗
StorJ	✗	✗	✗	✓	✓	✗
MaidSafe	✗	✗	✗	✓	✓	✗
Oasis	?	✓	✗	✗	✓	✗
Enigma	?	✓	✗	✗	✓	✗
Golem	?	✓	✗	✗	✓	✗
Blockstack	✓	✗	✗	✓	✓	✗
Truebit	?	✗	✗	✗	✗	✗
AUTOMATA	✓	✓	✓	✓	✓	✓

Table 1: Comparison to other systems or projects.

Comparison to existing solutions. In Table 1, IPFS [9] is the leading project that enables peer-to-peer file and resources sharing for blockchain users. Storj [8], MaidSafe [57] and other systems provide similar services but with different algorithms. These solutions allow end users to encrypt their data and upload to the platform to ensure the confidentiality of the data. However, they do not provide a comprehensive privacy-preserving mechanism to protect users' data privacy, especially data access patterns. At the same time, they are more focused on the data decentralization, but not confidential and privacy-first compute providers. Besides decentralized storage, Blockstack [58] also provides decentralized computation but without confidentiality. Truebit [59] presents a protocol to assist Ethereum smart contracts to outsource heavy computational tasks to off-chain nodes via a secure way, but it does not ensure the confidentiality and privacy of the code and data. Oasis [23], Enigma [22] and Golem [60] all embrace SGX and leverage its primitives to offer a confidential execution environment for smart contracts, which amplifies the smart contract's capabilities. Nevertheless, they are not designed to provide a high degree of privacy for the computation and storage, especially P2P storage. Since these systems are permissionless to the public to join as nodes, they also open the door for malicious nodes to sniff and infer the private information of their hosted users. When taking frontrunning into consideration, all of the existing protocols do not provide a comprehensive solution to eliminate the frontrunning attacks, and few of them offer mitigations but still leak access patterns to the unknow host nodes. To some extent, privacy of AUTOMATA is a must for all decentralized systems or applications. Compared to the existing approaches, AUTOMATA is the first system to provide privacy-first feature for both compute and storage planes, which is also frontrunning-resistant.

5 Conclusion

Privacy is always one of the primary concerns when users move their data from a centralized entity to a decentralized system. The vast majority of the present blockchain systems are acting as decent global distributed ledger systems but are not designed to resolve the privacy matters in a decentralized way. Therefore, developers could not build versatile Dapps on the current blockchain systems, which can compete with web or mobile applications. In this work, we provide AUTOMATA, a privacy-first infrastructure for Dapps, which enables developers to build any existing web/mobile apps or new apps in a decentralized and frontrunning-resistant manner.

References

[1] "Google," https://www.google.com, Accessed: 2020.

[2] "Facebook," https://www.facebook.com/, Accessed: 2020.

[3] "Amazon," https://www.amazon.com/, Accessed: 2020.

[4] "Facebook–cambridge analytica data scandal," https://en.wikipedia.org/wiki/Facebook\OT1\textendashCambridge_Analytica_data_scandal, Accessed: 2020.

[5] "Facebook says data leak hits 87 million users, widening privacy scandal," https://www.reuters.com/article/us-facebook-privacy/facebook-says-data-leak-hits-87-million-users-widening-privacy-scandal-idUSKCN1HB2CM , Accessed: 2020.

[6] "Everything you need to know about facebook's data breach affecting 50m users," https://techcrunch.com/2018/09/28/everything-you-need-to-know-about-facebooks-data-breach-affecting-50m-users/ , Accessed: 2020.

[7] "Ethereum," https://www.ethereum.org/, Accessed: 2020.

[8] "Storj.io," http://storj.io/, Accessed: 2020.

[9] "Ipfs," https://ipfs.io/, Accessed: 2020.

[10] "Uniswap," https://uniswap.exchange/, Accessed: 2020.

[11] “Curve,” https://www.curve.fi/, Accessed: 2020.

[12] “Maker dao,” https://makerdao.com/, Accessed: 2020.

[13] “Aave,” https://aave.com/, Accessed: 2020.

[14] “Ethereum users now have more than $10 billion at play in defi,” https://decrypt.co/42976/ethereum-10-billion-total-value-locked-defi, Accessed: 2020.

[15] “Ethereum is a dark forest,” https://medium.com/@danrobinson/ethereum-is-a-dark-forest-ecc5f0505dff, Accessed: 2020.

[16] “In an ethereum-based ’initial dex offering,’ open doesn’t always mean fair,” https://decrypt.co/35579/in-an-ethereum-based-initial-dex-offering-open-doesnt-always-mean-fair, Accessed: 2020.

[17] “Zcash,” https://z.cash/, Accessed: 2020.

[18] M. Blum, P. Feldman, and S. Micali, “Non-interactive zero-knowledge and its applications,” in *Proceedings of the Twentieth Annual ACM Symposium on Theory of Computing*, 1988.

[19] J. Groth, “Short non-interactive zero-knowledge proofs,” in *Advances in Cryptology - ASIACRYPT 2010*, M. Abe, Ed., 2010.

[20] H. Lipmaa, “Progression-free sets and sublinear pairing-based non-interactive zero-knowledge arguments,” in *Theory of Cryptography*, R. Cramer, Ed., 2012.

[21] E. Ben-Sasson, A. Chiesa, E. Tromer, and M. Virza, “Succinct non-interactive zero knowledge for a von neumann architecture,” in *Proceedings of the 23rd USENIX Conference on Security Symposium*, 2014.

[22] “Enigma,” https://www.oasislabs.com/, Accessed: 2020.

[23] “Oasis labs,” https://www.oasislabs.com/, Accessed: 2020.

[24] A. Kosba, A. Miller, E. Shi, Z. Wen, and C. Papamanthou, “Hawk: The blockchain model of cryptography and privacy-preserving smart contracts,” in *2016 IEEE Symposium on Security and Privacy (SP)*, 2016.

[25] G. Zyskind, O. Nathan, and A. . Pentland, “Decentralizing privacy: Using blockchain to protect personal data,” in *2015 IEEE Security and Privacy Workshops*, 2015.

[26] C. Liu, L. Zhu, M. Wang, and Y.-a. Tan, “Search pattern leakage in searchable encryption: Attacks and new construction,” *Information Sciences*, vol. 265, pp. 176–188, 2014.

[27] M. S. Islam, M. Kuzu, and M. Kantarcioglu, “Access pattern disclosure on searchable encryption: Ramification, attack and mitigation.” in *NDSS*, vol. 20, 2012, p. 12.

[28] Y. Jia, T. Moataz, S. Tople, and P. Saxena, “Oblivp2p: An oblivious peer-to-peer content sharing system,” in *Proceedings of the 25th USENIX Conference on Security Symposium*, 2016.

[29] S. Tople, Y. Jia, and P. Saxena, “Pro-oram: Constant latency read-only oblivious ram,” in *Proceedings of the 22nd International Symposium on Research in Attacks, Intrusions and Defenses*, 2019.

[30] Y. Jia, S. Tople, T. Moataz, D. Gong, P. Saxena, and Z. Liang, “Robust synchronous p2p primitives using sgx enclaves,” 2020.

[31] “Software Guard Extensions Programming Reference.” software.intel.com/sites/default/files/329298-001.pdf, Sept 2013.

[32] “Trust zone,” https://developer.arm.com/ip-products/security-ip/trustzone, Accessed: 2020.

[33] “Scaneye’s global bittorrent monitor,” http://www.cogipas.com/anonymous-torrenting/torrent-monitoring/.

[34] K. Bauer, D. McCoy, D. Grunwald, and D. Sicker, "Bitstalker: Accurately and efficiently monitoring bittorrent traffic," in *WIFS*, 2009.

[35] T. Chothia, M. Cova, C. Novakovic, and C. G. Toro, "The unbearable lightness of monitoring: Direct monitoring in bittorrent," in *SECURECOMM*, 2012.

[36] O. Goldreich and R. Ostrovsky, "Software protection and simulation on oblivious rams," *J. ACM*, 1996.

[37] E. Shi, T.-H. Chan, E. Stefanov, and M. Li, "Oblivious RAM with $O(\log^3(N))$ Worst-Case Cost," in *ASIACRYPT*, 2011.

[38] E. Stefanov, M. van Dijk, E. Shi, C. Fletcher, L. Ren, X. Yu, and S. Devadas, "Path ORAM: an extremely simple oblivious RAM protocol," in *CCS*, 2013.

[39] L. Ren, C. Fletcher, A. Kwon, E. Stefanov, E. Shi, M. van Dijk, and S. Devadas, "Constants Count: Practical Improvements to Oblivious RAM ," 2014.

[40] S. Devadas, M. van Dijk, C. Fletcher, L. Ren, E. Shi, and D. Wichs, "Onion ORAM: A Constant Bandwidth Blowup Oblivious RAM," *IACR Cryptology ePrint Archive*, 2015.

[41] T. Moataz, T. Mayberry, and E.-O. Blass, "Constant Communication ORAM with Small Blocksize," in *CCS*, 2015.

[42] D. Dachman-Soled, C. Liu, C. Papamanthou, E. Shi, and U. Vishkin, "Oblivious network RAM and leveraging parallelism to achieve obliviousness," in *ASIACRYPT*, 2015.

[43] E. Boyle, K. Chung, and R. Pass, "Oblivious parallel RAM and applications," in *TCC*, 2016.

[44] E. Stefanov, E. Shi, and D. X. Song, "Towards practical oblivious ram," in *NDSS*. The Internet Society, 2012.

[45] E. Stefanov, M. van Dijk, E. Shi, C. W. Fletcher, L. R. 0001, X. Yu, and S. Devadas, "Path oram: an extremely simple oblivious ram protocol," in *2013 ACM SIGSAC Conference on Computer and Communications Security*, 2013.

[46] S. Sasy, S. G. 0001, and C. W. Fletcher, "Zerotrace: Oblivious memory primitives from intel sgx," in *NDSS*. The Internet Society, 2018.

[47] "Amazon web services," https://aws.amazon.com/, Accessed: 2020.

[48] "Google cloud," https://cloud.google.com/, Accessed: 2020.

[49] "Google docs," https://www.google.com/docs/, Accessed: 2020.

[50] "Slack," https://slack.com, Accessed: 2020.

[51] S. Tople, S. Park, M. S. Kang, and P. Saxena, "Vericount: Verifiable resource accounting using hardware and software isolation," in *Applied Cryptography and Network Security - 16th International Conference*, 2018.

[52] G. Ateniese, R. C. Burns, R. Curtmola, J. Herring, L. Kissner, Z. N. J. Peterson, and D. X. Song, "Provable data possession at untrusted stores," in *Proceedings of the 2007 ACM Conference on Computer and Communications Security*, 2007.

[53] A. Juels and B. S. K. Jr, "Pors: proofs of retrievability for large files," in *Proceedings of the 2007 ACM Conference on Computer and Communications Security*, 2007.

[54] K. Ranganathan, A. Iamnitchi, and I. Foster, "Improving data availability through dynamic model-driven replication in large peer-to-peer communities," in *2nd IEEE/ACM International Symposium on Cluster Computing and the Grid (CCGRID'02)*, 2002.

[55] Z. Zhang and Q. Lian, "Reperasure: Replication protocol using erasure-code in peer-to-peer storage network," in *21st IEEE Symposium on Reliable Distributed Systems, 2002. Proceedings.*, 2002.

[56] M. Luby, M. Mitzenmacher, M. A. Shokrollahi, D. A. Spielman, and V. Stemann, "Practical loss-resilient codes," in *STOC*, 1997.

[57] "Maidsafe," https://maidsafe.net/, Accessed: 2020.

[58] "Blockstack," https://blockstack.org/, Accessed: 2020.

[59] "Truebit protocol," https://truebit.io/, Accessed: 2020.

[60] "Golem," https://golem.network/, Accessed: 2020.