ETH zürich

ETH Library

Towards Blockchain-based Auditable Storage and Sharing of IoT Data

Conference Paper

Author(s):
Shafagh, Hossein; Burkhalter, Lukas; Hithnawi, Anwar; Duquennoy, Simon

Publication date:
2017-11-03

Permanent link:
https://doi.org/10.3929/ethz-b-000218655

Rights / license:
In Copyright - Non-Commercial Use Permitted

Originally published in:
https://doi.org/10.1145/3140649.3140656

This page was generated automatically upon download from the ETH Zurich Research Collection.
For more information, please consult the Terms of use.

Towards Blockchain-based Auditable Storage and Sharing of IoT Data

Hossein Shafagh
Department of Computer Science
ETH Zurich, Switzerland
shafagh@inf.ethz.ch

Lukas Burkhalter
Department of Computer Science
ETH Zurich, Switzerland
burkhalter@inf.ethz.ch

Anwar Hithnawi
Department of Computer Science
ETH Zurich, Switzerland
hithnawi@inf.ethz.ch

Simon Duquennoy
RISE SICS, Sweden
simon.duquennoy@ri.se

ABSTRACT

Today the cloud plays a central role in storing, processing, and distributing data. Despite contributing to the rapid development of IoT applications, the current IoT cloud-centric architecture has led into a myriad of isolated data silos that hinders the full potential of holistic data-driven analytics within the IoT. In this paper, we present a blockchain-based design for the IoT that brings a distributed access control and data management. We depart from the current trust model that delegates access control of our data to a centralized trusted authority and instead empower the users with data ownership. Our design is tailored for IoT data streams and enables secure data sharing. We enable a secure and resilient access control management, by utilizing the blockchain as an auditable and distributed access control layer to the storage layer. We facilitate the storage of time-series IoT data at the edge of the network via a locality-aware decentralized storage system that is managed with the blockchain technology. Our system is agnostic of the physical storage nodes and supports as well utilization of cloud storage resources as storage nodes.

1 INTRODUCTION

With the emergence of networked embedded devices dubbed as the IoT, we are witnessing an ever increasing number of innovative applications. The current ecosystem of the IoT consists typically of designated low-power devices equipped with sensors collecting data. This data is then stored via special-purpose apps (i.e., application-layer gateways) in a third-party cloud storage for further processing.

This stove-piped architecture [31] has resulted into isolated data silos, where users have limited control over their data and how it is used. Users have to trust the cloud and application providers and have no choice but to rely on their promises of security and availability.

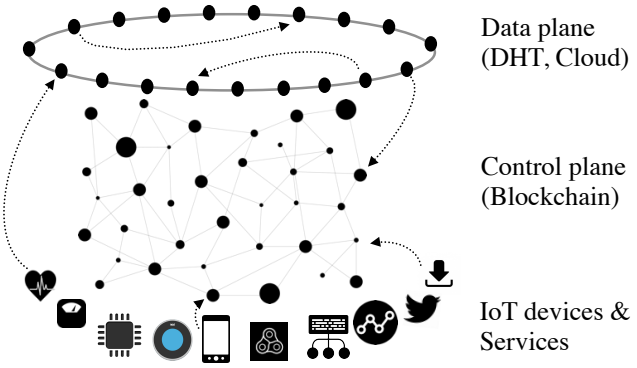

Figure 1: Our blockchain-based, end-to-end encrypted, and decentralized data storage. No central trusted authority controls access to users' data.

Today's IoT security efforts mostly focus on securing point-to-point communication and fall short in addressing security during the life-cycle of data (e.g., auditable access control, secure sharing). The current cloud-based model in the IoT handles identification, authentication, and connectivity of IoT devices. Although this model has enabled the bootstrap of the IoT ecosystem, it is not necessarily the most suitable solution for the IoT [32], as it neglects the locality of data and mandates centralization through trusted third parties.

These limitations necessitate a rethinking of the way we currently handle IoT data. Instead of giving up ownership of our data to various applications, we enable an independent and resilient data management system that ensures data ownership. We identify the following requirements for such a system: *(R1)* Decentralized, resilient, and auditable access control management (ownership, cryptographically secure sharing); *(R2)* Secure data storage (confidentiality, authenticity, integrity); *(R3)* IoT compatibility (append-only data streams, with a single writer and several readers). Conventional cloud-based solutions provide *R2* and when relevant *R3*, but fall short in addressing *R1*. Recent decentralized storage startup efforts (Sia [25], Storj [29], Filecoin [27]) show the potential of decentralized blockchain-based storage with financial incentives.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than the author(s) must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.
CCSW'17, November 3, 2017, Dallas, TX, USA
© 2017 Copyright is held by the owner/author(s). Publication rights licensed to ACM.
ACM 978-1-4503-5204-8/17/11…$15.00
https://doi.org/10.1145/3140649.3140656

These efforts, however, are optimized for file storage and fall short in accommodating for time-series data of the IoT (i.e., *R3*).

We propose a blockchain-based access control management to address *R1*. This provides us with an independent network that maintains a distributed ledger of access control permissions. Inspired by recent blockchain-based technologies [2, 33], we combine the blockchain with an off-chain storage, for a scalable secure data storage to address *R2*. Finally, our system is designed from the ground up to support IoT data streams to address *R3*. Our system accommodates for IoT data streams where streams are chunked, compressed, and encrypted in the application layer and only authorized services are granted access to the decryption keys. This requires us to address several challenges for an efficient key distribution and management scheme as well as a secure storage of data stream chunks. The built-in cryptocurrency feature in the underlying blockchain technology would allow the realization of an autonomous, self-sustaining decentralized storage ecosystem, where storage nodes are rewarded for providing storage and bandwidth, and more importantly for following the protocol correctly.

The contribution of this paper is the design of a novel blockchain-based auditable data-management system for IoT data. Our system exhibits *(i)* a cryptographically secure data sharing with frequent key updates, *(ii)* the possibility of access revocation, *(iii)* an efficient search of compressed chunked data streams, and *(iv)* a distributed locality-aware storage layer.

2 BACKGROUND

In this section, we briefly review important aspects of the cloud, the IoT ecosystem, and give a primer on the blockchain technology.

2.1 Cloud

Cloud platforms are typically hosted in large-scale data centers that are located at the edge of the Internet backbone [32]. The consolidation and centralization of data centers, however, yield an increased distance between clients and services. This results in a high variability in latency and bandwidth. To address this issue, especially with regards to resource-intensive and interactive applications, decentralized cloud architectures, namely cloudlets, have emerged. Cloudlets are small-scale data centers that are located closer to users and can meet low latency and high bandwidth guarantees. Our system embraces this locality-aware data storage and processing trend and brings it to its full potential with our decentralized access control layer which ensures ownership and secure sharing of data.

2.2 IoT Ecosystem

Embedded computing devices are increasingly integrated into objects and environments surrounding us. These devices utilize low-cost sensors for a range of applications. The typical system structure of the IoT involves the three tiers of (i) low-power IoT devices, (ii) a potential gateway that interconnects IoT devices with the Internet, and (iii) the backend where IoT data is stored.

IoT devices are typically equipped with resources in the orders of few MHz of CPU, few 10s of KB of RAM, and few 100s of KB of ROM. Additionally, they can embed low-power hardware crypto accelerators, enabling a new class of secure applications [23, 24], for instance, lightweight clients of a blockchain network. However, conventional security solutions for the IoT still utilize pre-shared symmetric keys for the secure communication. This simple approach does not scale for the massive number of IoT devices. Efforts [14] to tailor public-key based secure communication to the IoT remain to find widespread adoption. Leveraging the blockchain technology, we enable a decentralized management of identities of IoT devices and enable a transparent device ownership.

2.3 Blockchain

A blockchain is essentially a distributed ledger that consists of a continuously growing set of records. The distributed nature of blockchains implies no single entity controls the ledger (i.e., censorship/suppression resistant), but rather the participating peers together validate the authenticity of records. These records are organized in blocks which are linked together using cryptographic hashes, hence the name blockchain. Blockchain-based technologies [19] incentivize a network of peers to make computations towards consensus in the network.

The most prominent example of a successful blockchain deployment is the Bitcoin cryptocurrency (the decentralized peer-to-peer digital currency) [6, 18]. The Bitcoin blockchain maintains all transactions from the initial block, referred to as the genesis block. A transaction contains the sender, receiver, amount of the transferred Bitcoin currency, and signature of the sender. For a transaction to be included in the blockchain (i.e., to be considered as valid), it is transmitted to the blockchain network. The so-called miners take the responsibility to verify new transactions and suggest the next block which includes the verified transactions. Miners are rewarded with Bitcoins and transaction fees for their computational work.

To prevent a single miner from dominating the blockchain network and hence having the power of manipulating the history of transactions, the concept of proof-of-work [18] is employed to reach consensus in the blockchain network. A new block includes a set of new transactions, the hash of the previous block, the miner's address who is suggesting this block, and most importantly the answer to a difficult-to-solve mathematical puzzle. This mathematical puzzle is unique to each block and easy to verify once found. Once a miner finds such a block, it publishes it such that all nodes and miners can verify its correctness and consider it as the new valid block to build upon. In case several valid blocks are suggested at the same time, miners randomly select the next block. Eventually, the network converges towards the longest branch of the blockchain as the main branch. Solving the puzzle is referred to as the proof-of-work and ensures as well resistance against Sybil attacks.

Bitcoin and its most prominent contender Ethereum [1] are permission-less blockchains where any node can become a miner or just a client. Permissioned blockchains, such as the hyperledger [7], allow a designated set of authorized validator nodes (i.e., miners) to participate in the block validation process. Such blockchains typically use more CPU-friendly consensus protocols, such as the Practical Byzantine Fault Tolerance protocol [8], since the set of validator nodes is known. Hence, permissioned blockchains can handle a higher transaction throughput (7 vs. 10^4 transactions per second). However, permissioned blockchains require a trusted central party to initially authorize the blockchain validators.

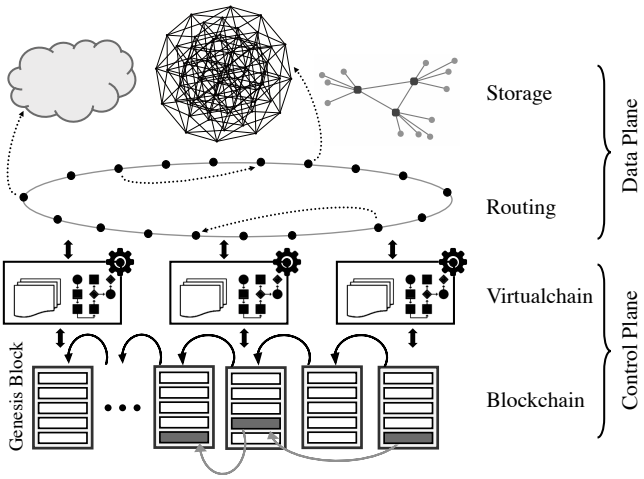

Figure 2: Overview of our layered design. Transactions in the blockchain can contain access permissions (gray).

Moreover, due to the high communication overhead, i.e., $O(n^2)$, only deployments of a few tens of validators are practical.

3 SYSTEM DESIGN

In a nutshell, we decouple the control and data plane of our IoT distributed storage system (see Fig. 2). We realize the access control layer using a public blockchain, to satisfy *R1*. Bitcoin is our current candidate for the blockchain layer in our reference implementation due to its strong security, reliability, and current dominance. However, other cryptocurrencies [22] can be employed seamlessly. This is possible, because our system's logic resides in a virtualchain [2, 21] and outside the blockchain. Virtualchain allows the introduction of new functionality to production blockchains, without requiring any changes in the underlying blockchain.

The data plane consists of a routing layer and the secure storage layer to satisfy *R2*. The storage layer is composed of either an on-premises storage, the cloud, or a distributed peer-to-peer network. Data is encrypted end-to-end at the client-side. Hence, the storage nodes have no insights about the hosted data at their side. The data in our system is structured in streams, to accommodate for IoT-specific needs (statisfies *R3*). In concrete terms, ownership and sharing permissions are per stream, and streams are chunked and encrypted before storage.

In the following, we detail our design for the control plane and our data plane features.

3.1 Control Plane

In our system, the control plane is logically separated and agnostic of the data plane.

Blockchain. We employ a publicly verifiable blockchain to create an accountable distributed system and bootstrap trust in an untrusted network, without a central trust entity. In our system, transactions consist of ownership of data streams and corresponding access permissions. Our access control transactions, similar to default transactions of the underlying cryptocurrency, remain publicly auditable (see Fig. 2). To preserve the privacy of access permissions, we can rely on stealth addresses [9].

Access Control. We use the blockchain to store access permissions securely. Access rights are granted per data stream and the data owner can revoke the sharing of a data stream. Initially, the data owner issues a transaction including the stream identifier (i.e., hash digest). To share the data stream with a service, the data owner issues a new transaction which holds *(i)* the stream identifier and *(ii)* the public key address of the service.

For any request to retrieve data, the storage node first checks the blockchain for access rights. Note that a malicious storage node could hand out data without permission. However, the impact of this action is limited since *(i)* data is encrypted, *(ii)* in the case of DHT, each node holds a small random fraction of a data stream. Moreover, economic incentives (i.e., collateral and reward) should encourage storage nodes to follow the protocol correctly.

Key Management. We enable a low-cost key renewal with key regression [12]. In key regression, given key K_t in current time t one can compute all keys until the initial key K_0. This allows us to update the encryption keys frequently, and only share the latest K_t with the sharing services. However, given n services, this requires a communication overhead in the order of $O(n)$: at each key update, the key must be shared n times (after encrypting it with the corresponding service's public key).

We propose to employ a re-encryption-based technique to bring the communication overhead to $O(1)$. Given a re-encryption token $T_{a\rightarrow b}$, one can re-encrypt a ciphertext under Alice's public key PK_a to a ciphertext under Bob's public key PK_b, without access to the plaintext [3, 5]. To share K_t with all services, Alice encrypts K_t with a one-time public key pair (PK_a, SK_a). For all services S_i, she issues a re-encryption token $T_{SK_a\rightarrow PK_{S_i}}$ based on their public keys PK_{S_i} (this step takes place while issuing the sharing transaction). Each service S_i can then re-encrypt $\text{ENC}_{PK_a}(K_t)$ to $\text{ENC}_{PK_{S_i}}(K_t)$, and use their respective secret keys (SK_{S_i}) to access K_t. After this point, Alice only needs to update $\text{ENC}_{PK_a}(K_{t+1})$ for the services to preserve their access to the latest key K_{t+1}.

Revocation. To revoke access to a data stream, the data owner updates the encryption key to K_{t+1}. She then updates the encrypted shared key for authorized readers, however, with a new one-time public key pair: $\text{ENC}_{PK_{a'}}(K_{t+1})$. Revocation causes a communication overhead of $O(n)$, since Alice needs to update all valid re-encryption tokens $T_{SK_{a'}\rightarrow PK_{S_i}}$, excluding the revoked service. This prevents the revoked user to decrypt any future data.

As an additional protection, and for auditing purposes, the user issues a new blockchain transaction overriding previous permissions. Storage nodes will even decline sharing older data, that the user once had access to. The impact of a potential dishonest node leaking old encrypted data chunks is low, as old data might have been cached at the user anyway. New data, however, is protected cryptographically after a key update.

3.2 Data Plane

In order to address *R3*, we consider IoT data types of stream character where data records are generated continuously, as depicted in Fig. 3. Current distributed storage approaches [15, 27, 29] primarily target archiving data and are not suitable for IoT data.

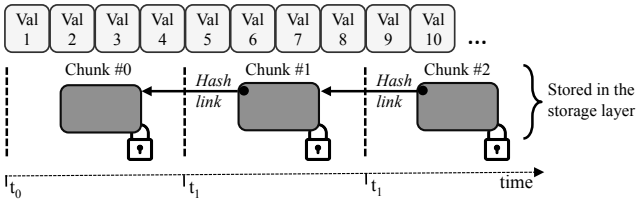

Figure 3: Data streams are chunked at pre-defined lengths, compressed, and encrypted. To lookup a record, a local index maps the key of the record to the chunk key.

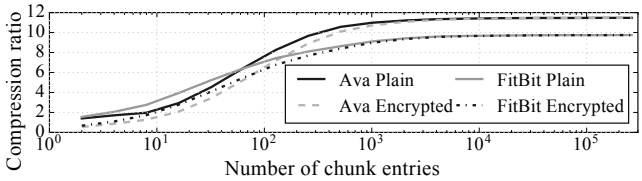

Figure 4: Compression ratio of our chunking with the authors Fitbit data and anonymized Ava data of one year.

Moreover, they either consider data to be public (e.g., IPFS[15]) or store encrypted data without a secure sharing feature (e.g., Storj [29] and Filecoin [27]).

To store time-series data in our system, we store data chunks which compose several consecutive data records, instead of storing individual data records. To this end, we split a data stream into data chunks which are cryptographically chained together (i.e., each chunk holds a hash pointer to the previous chunk). Although chunking data prevents random access at the record level, there is a positive gain on the performance of data retrieval since in time-series data most queries require data that is co-located in time (e.g., all records of one day) [13].

Note that the data itself is stored off-chain and only its identifier (i.e., hash pointer) is included in the blockchain, ensuring data immutability. Since adding an identifier for each chunk to the blockchain would not scale, our system adds only data chunks at given intervals into the blockchain. Due to the fact that all chunks are cryptographically chained together, all chunks that are between two intervals without their identifier in the blockchain become immutable too. The interval-time corresponds to the maximum time chunks need to become immutable. It is tunable and defined by the application logic.

Encryption. Each data chunk is encrypted at the source with an efficient symmetric cipher. We rely on AES-GCM, as an authenticated encryption scheme. Our chunks have a plaintext field containing the key value of the chunk and the encrypted compressed data records. With authenticated encryption, both fields are integrity protected and authenticated. Services with access to the symmetric data stream key K_t can verify the integrity of the chunk and perform an authenticated decryption.

To ensure data ownership, for instance towards the storage layer, each chunk is in addition signed. This allows parties without access to the stream key to still be able to verify the owner of the data stream, albeit at a higher computation cost. Each chunk contains the unencrypted stream identifier linking it to the corresponding access control transactions.

Compression. IoT data is highly compressible due to high correlation in time. Hence, we compress data chunks before encryption. This reduces bandwidth and storage requirements significantly. Our initial results of IoT data compression show that even with small chunk sizes, we can reach compression ratios close to the optimum (i.e., compression of the entire data set). As depicted in Fig. 4, compressing the data record of one year by Fitbit[1] results into a compression ratio of 9.75 (11.45 for Ava[2]). Already with a chunk size of 2048 (corresponding to one day worth of data records for Ava), we can reach a ratio of 11.08 for encrypted and compressed chunks.

Search. In the storage layer, we store key-value pairs. In our case, the value is the current data chunk of a data stream, where the key (i.e., a 256-bit identifier) is the cryptographic hash of the tuple: <stream-ID, owner-ID, timestamp-hash>. The IDs are unique bit strings (i.e., hash digests).

To enable an efficient search and query of any record in the data stream, we use a simple technique based on the the timestamp t_0 of the first chunk and the length of the chunks Δ. To look-up a record with timestamp t_i, we compute the timestamp of the chunk holding it. For instance, the look-up of value 6 in Fig. 3 is mapped to the key of chunk #1.

Data Storage. We advocate a distributed data storage layer, however our design is agnostic of the storage layer. Hence, on-premise storage and storage on cloud services are compatible with our system.

The IoT gateway serves as an intermediate storage node at the front of the storage layer. The gateway can push the chunks in a FIFO principle into the storage layer to maintain a reasonable local storage size. Hence, the IoT gateway serves as a cache of recent data items for on the gateway hosted apps and could as well be queried by services in network proximity to the IoT gateway. Current IoT apps, first push all new data to the cloud and then fetch them again for presentation (e.g., FitBit), introducing an unnecessary latency.

For the distributed storage, we rely on a P2P overlay routing technique and a *Distributed Hash Table* (DHT) as our general-purpose private key-value data store interface. The DHT serves as a scalable, self-managing storage with high availability (i.e., robust against targeted communication outages or malicious attacks in case of central servers). The DHT in general enforces replicated and randomized storage across a 160-bit address space. However, we rely on the sloppy hashing technique [10, 11] to augment our instantiation of DHT [4, 17] with locality. To this end, chunks are stored/replicated on nodes that are closer to the services or the data owner.

How to financially incentivize participation in a distributed storage is out of the scope of this paper. Several researchers and a few start-ups propose systems where users can make money with "renting out" their local storage space [27–29]. For instance, proof-of-retrievability [29] can be used as a mechanism to reward users

[1] Fitbit heart rate & fitness wirstband: www.fitbit.com/charge2
[2] Ava: ovulation tracking bracelet: www.avawomen.com

who store more files for a longer time. Moreover, the reward encourages storage nodes to follow the protocol correctly, for example with regards to enforcing the access permissions.

3.3 Privacy & Security Analysis

For an adversary to alter access permissions in the blockchain it requires forging a digital signature or gaining control over the majority of the compute power in the blockchain network. The former is prevented with the security of signatures and the latter with the consensus protocol in the blockchain (i.e., proof-of-work) and its decentralized nature. Moreover, an adversary is not capable of learning sensitive information from the public blockchain, since only pseudo-identities and stream identifiers are stored there. Data chunks are encrypted, integrity protected, and authenticated. An adversary with access to encryption keys cannot alter stored chunks, as it requires gaining access to the public-private key pair of the data owner. Even in case the owner's key is leaked, chunks cannot be modified due to the blockchain immutability (except for chunks in the current interval). Rational storage nodes follow the protocol correctly due to financial incentives (i.e., interplay of reward and collateral)

3.4 Primary Evaluation

Initial evaluation results from our reference implementation in the bitcoin blockchain (i.e., bitcoin testnet) show reasonable overhead. For instance, augmenting Amazon's S3 storage with our system's access control results only in a slowdown of 10% in request throughput. In comparison, a distributed storage with more than 1000 nodes experiences a factor of 2 slowdown, dominated by the routing. This slowdown is the worst-case scenario with no locality-awareness in storage of data. Store and get procedures for individual chunks require 150 ms, assuming no local caching. We are currently working on thoroughly evaluating our system and analyzing the performance of several real-world IoT applications on top of it.

4 RELATED WORK

In this section, we briefly review a subset of relevant work to our system.

Data Privacy & Access Control. The current dominating approach for sharing data in the web services is based on the OAuth protocol [16] where a central trusted entity enforces user-defined access policies (does not fullfil *R1*). Sieve [30] addresses these shortcomings with a combination of key-homomorphic and attribute-based encryption schemes. Many applications employ anonymized data collection [26] as an attempt to protect personally identifiable information. However, researchers have demonstrated effective de-anonymization techniques [20] which work even with a small set of high dimensionality data.

Blockchain. In recent years, a new class of blockchain technologies have emerged that utilize the accountable computing and auditability of blockchains for other domains. Blockstack [2] introduces the concept of virtualchains and proposes a decentralized server-less DNS. Blockstack extends to a decentralized public key distribution system and registry for user identities. Storj [29] and FileCoin [27] (secure successor of IPFS [15]) introduce a distributed object storage. They are both targeted for archiving files and lack sharing features. Enigma [33, 34] is the closest to our approach in that it uses the blockchain for access control and enables sharing of off-chain stored data. However, Enigma stores data access logs within the blockchain, without addressing the consequential scalability issues. Moreover, their system does not accommodate for IoT stream data (not satisfing *R3*). Our approach is inspired by the above approaches, however, our focus on IoT data leads to a number of important design differences.

IoT Storage. A few of our design decisions regarding IoT data streams are inspired by Bolt [13]. Bolt presents chunking of IoT data for performance gain and protects confidentiality of chunks. However, Bolt relies on the cloud-centric model (does not address *R1*). "The Cloud is Not Enough" [32] discusses the pitfalls of the cloud-centric IoT and advocates a data-centric approach. They leave concrete system proposals for future work.

5 CONCLUSION

In this paper, we introduce the primary design of a distributed secure data storage system targeted for the Internet of Things. Our system allows for fine-grained access control and sharing of time-series sensor data of various IoT applications. Initial performance evaluation results are promising and show a moderate overhead due to our system. We are currently in the process of finalizing a complete reference implementation of our system and building several IoT applications on top of it.

REFERENCES

[1] Ethereum White-Paper. Online: https://github.com/ethereum/wiki/wiki/White-Paper, 2017.

[2] Ali, M., Nelson, J., Shea, R., and Freedman, M. J. Blockstack: A Global Naming and Storage System Secured by Blockchains. In *USENIX ATC* (2016).

[3] Ateniese, G., Fu, K., Green, M., and Hohenberger, S. Improved Proxy Re-encryption Schemes with Applications to Secure Distributed Storage. In *Symposium on Network and Distributed System Security (NDSS)* (2005).

[4] Baumgart, I., and Mies, S. S/Kademlia: A Practicable Approach Towards Secure Key-based Routing. In *IEEE International Conference on Parallel and Distributed Systems* (2007).

[5] Blaze, M., Bleumer, G., and Strauss, M. Divertible Protocols and Atomic Proxy Cryptography. In *EUROCRYPT* (1998).

[6] Bonneau, J., Miller, A., Clark, J., Narayanan, A., Kroll, J. A., and Felten, E. W. SoK: Research Perspectives and Challenges for Bitcoin and Cryptocurrencies. In *IEEE Symposium on Security and Privacy* (2015).

[7] Cachin, C. Architecture of the Hyperledger Blockchain Fabric. In *Workshop on Distributed Cryptocurrencies and Consensus Ledgers* (2016).

[8] Castro, M., and Liskov, B. Practical Byzantine Fault Tolerance and Proactive Recovery. *ACM Transactions on Computer Systems (TOCS) 20*, 4 (2002), 398–461.

[9] Courtois, N. T., and Mercer, R. Stealth Address and Key Management Techniques in Blockchain Systems. In *ICISSP* (2017).

[10] Freedman, M. J., Freudenthal, E., and Mazieres, D. Democratizing Content Publication with Coral. In *USENIX NSDI* (2004).

[11] Freedman, M. J., and Mazieres, D. Sloppy Hashing and Self-Organizing Clusters. In *International Workshop on Peer-to-Peer Systems* (2003).

[12] Fu, K., Kamara, S., and Kohno, T. Key Regression: Enabling Efficient Key Distribution for Secure Distributed Storage. In *Symposium on Network and Distributed System Security (NDSS)* (2006).

[13] Gupta, T., Singh, R. P., Phanishayee, A., Jung, J., and Mahajan, R. Bolt: Data Management for Connected Homes. In *USENIX NSDI* (2014).

[14] Hummen, R., Shafagh, H., Raza, S., Voig, T., and Wehrle, K. Delegation-based Authentication and Authorization for the IP-based Internet of Things. In *IEEE International Conference on Sensing, Communication, and Networking (SECON)* (2014).

[15] Juan Benet. IPFS - Content Addressed, Versioned, P2P File System (DRAFT 3). https://github.com/ipfs/papers, 2017.

[16] Lodderstedt, T., McGloin, M., and Hunt, P. OAuth 2.0 Threat Model and Security Considerations. *IETF, RFC 6819* (January 2013).

[17] Maymounkov, P., and Mazieres, D. Kademlia: A Peer-to-Peer Information System based on the XOR Metric. In *International Workshop on Peer-to-Peer Systems* (2002), pp. 53–65.

[18] Nakamoto, S. Bitcoin: A Peer-to-Peer Electronic Cash System, 2008.
[19] Narayanan, A., Bonneau, J., Felten, E., Miller, A., and Goldfeder, S. *Bitcoin and Cryptocurrency Technologies.* Princeton University Press, 2016.
[20] Narayanan, A., and Shmatikov, V. Robust de-anonymization of Large Sparse Datasets. In *IEEE Symposium on Security and Privacy* (2008).
[21] Nelson, J., Ali, M., Shea, R., and Freedman, M. J. Extending Existing Blockchains with Virtualchain. In *Workshop on Distributed Cryptocurrencies and Consensus Ledgers* (2016).
[22] Sasson, E. B., Chiesa, A., Garman, C., Green, M., Miers, I., Tromer, E., and Virza, M. Zerocash: Decentralized Anonymous Payments from Bitcoin. In *IEEE Symposium on Security and Privacy* (2014).
[23] Shafagh, H., Hithnawi, A., Burkhalter, L., Fischli, P., and Duquennoy, S. Secure Sharing of Partially Homomorphic Encrypted IoT Data. In *ACM Conference on Embedded Networked Sensor Systems (SenSys)* (2017).
[24] Shafagh, H., Hithnawi, A., Dröscher, A., Duquennoy, S., and Hu, W. Talos: Encrypted Query Processing for the Internet of Things. In *ACM Conference on Embedded Networked Sensor Systems (SenSys)* (2015).
[25] Sia. Sia: Decentralized Private Cloud. https://siawiki.tech/, 2017.
[26] Sweeney, L. k-anonymity: A Model for Protecting Privacy. *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems 10*, 05 (2002), 557–570.
[27] Techical Report. Filecoin: A Cryptocurrency Operated File Network. http://filecoin.io/filecoin.pdf, 2014.
[28] Techical Report. MaidSafe: Project Safe. https://github.com/maidsafe/Whitepapers, 2014.
[29] Techical Report. Storj: A Peer-to-Peer Cloud Storage Network. https://storj.io/storj.pdf, 2016.
[30] Wang, F., Mickens, J., Zeldovich, N., and Vaikuntanathan, V. Sieve: Cryptographically Enforced Access Control for User Data in Untrusted Clouds. In *USENIX NSDI* (2016).
[31] Zachariah, T., Klugman, N., Campbell, B., Adkins, J., Jackson, N., and Dutta, P. The Internet of Things Has a Gateway Problem. In *Proceedings of the 16th International Workshop on Mobile Computing Systems and Applications (HotMobile)* (2015).
[32] Zhang, B., Mor, N., Kolb, J., Chan, D. S., Lutz, K., Allman, E., Wawrzynek, J., Lee, E., and Kubiatowicz, J. The Cloud is Not Enough: Saving IoT from the Cloud. In *USENIX HotCloud* (2015).
[33] Zyskind, G., Nathan, O., and Pentland, A. Decentralizing Privacy: Using Blockchain to Protect Personal Data. In *IEEE Security and Privacy Workshops* (2015).
[34] Zyskind, G., Nathan, O., and Pentland, A. Enigma: Decentralized Computation Platform with Guaranteed Privacy. arXiv (whitepaper) http://www.enigma.co/enigma_full.pdf, 2015.