DL-BAC: Distributed Ledger Based Access Control for Web Applications

Lei Xu
University of Houston
Houston, Texas, 77204
xuleimath@gmail.com

Lin Chen
University of Houston
Houston, Texas, 77204
chenlin198662@gmail.com

Nolan Shah
University of Houston
Houston, Texas, 77204
nshah10@uh.edu

Zhimin Gao
University of Houston
Houston, Texas, 77204
mtion@hotmail.com

Yang Lu
University of Houston
Houston, Texas, 77204
ylu17@uh.edu

Weidong Shi
University of Houston
Houston, Texas, 77204
wshi3@uh.edu

ABSTRACT

Since Internet based applications have become the norm for most users, security has become a bigger concern than ever before, especially for applications like social networking and cloud based storage. Access control is one of the key techniques that can mitigate security concerns for web based applications. However, most existing access control mechanisms require a trusted party, which are vulnerable to many threats including malicious insiders and single point failure. In response to these challenges, we propose DL-BAC, a novel access control system based on the distributed ledger. DL-BAC robustly enforces access control policies without depending on a single trusted party. We also provide an extension of DL-BAC that is privacy respecting and evaluate the performance of DL-BAC to show its practicability.

Keywords

web application, security, distributed ledger

1. INTRODUCTION

The quick development of Internet and web technologies are creating unprecedented opportunities in areas such as e-commerce, social networking, and online storage. Security is critical for these web applications due to several factors, e.g., high volumes of sensitive information managed by web applications, interconnections of heterogeneous and distributed systems, and difficulties in detection and prosecution of anonymous/remote computer crimes [17].

Joshi et.al. classifies security services for web applications in two categories [17]. The first, secure communications, is provided through the comprehensive protection provided by the wide adoption of PKI [2] and advanced encryption AES [1]. The second, access control, has received a lot of attention [31, 29, 36, 5, 4, 40]; however, existing works focus on efficient access policy design and formal verification. For policy enforcement mechanism and auditability, most existing schemes assume a trusted party but ignore the scenario that there is no trusted party. For web applications like cloud-based storage and social networking, the situation is especially serious because the user has to rely on the service provider to enforce his/her access control policies. If a malicious insider (e.g., rogue system administrator) or an attacker compromises the server, they can easily bypass or disable the access control mechanism and retrieve user data without leaving any trace in the system.

To address these challenges for the access control mechanism of web applications, we propose a distributed ledger based access control system (DL-BAC) which does not rely on any single trusted party and improves security for web applications. As public accessibility is a key feature of the distributed ledger, privacy becomes a concern when all access control related information (e.g., access policies and access histories) are stored on the ledger and available to everyone. We leverage a zero-knowledge proof technique to mitigate this problem without disrupting access control functionalities. In summary, our contributions in this paper include:

- We design DL-BAC, a distributed ledger based access control system for different web applications without assuming any single trusted party;
- We provide a privacy protection extension of DL-BAC that hides access policies and access history stored on the distributed ledger from the public while supporting necessary access control operations;
- We analyze the security features of DL-BAC and evaluate its performance to show the practicality.

The remainder of the paper is organized as follows: Section 2 gives a short review of the distributed ledger technology. Section 3 provides a detailed description of the basic version of access control based on distributed ledger DL-BAC, and Section 4 a privacy enhanced DL-BAC is proposed. Section 5 evaluates the performance of the two proposed approaches and Section 6 discusses related work. We conclude the paper in Section 7.

2. BACKGROUND OF DISTRIBUTED LEDGER

The distributed ledger (blockchain) is at the core of the cryptocurrency, Bitcoin [24], and a major technology con-

WWW'17 Companion, April 3–7, 2017, Perth, Australia.
ACM 978-1-4503-4914-7/17/04.
http://dx.doi.org/10.1145/3041021.3053897

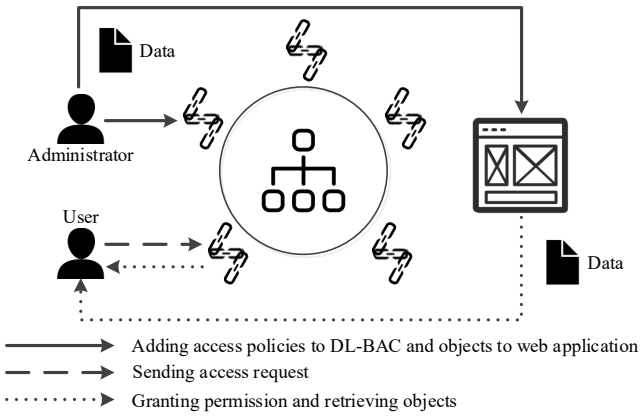

Figure 1: Overview of DL-BAC. The administrator of the web application stores encrypted data on the server and submits access policies together with data encryption key to the distributed ledger system. When a user tries to access the stored data, the request is processed by the distributed ledger to determine whether the request should be allowed or denied. If it is allowed, the distributed ledger distributes the data encryption key to the user in a secure way.

tributor to the success of Bitcoin over a community of distributed peers. It is believed that distributed ledger based technology may revolutionize many industries [37, 35].

Roughly speaking, a distributed ledger is a system involves multiple participants who achieve consensus on a data set and maintain the data locally. Distributed ledger systems are developed under different trust models with different consensus protocols. There are two main types of trust models: one assumes all participants are equivalent and one has participants with varying privileges for block construction. Under a given trust model, the system can use various consensus protocols such as proof-of-work [24], proof-of-stake [8], or Byzantine fault tolerance (BFT) [38]

Regardless of the trust model and underlying consensus protocol, most distributed ledger systems have three important features [26, 15]: (i) public accessibility (all information stored with block-chain is publicly accessible to everyone); (ii) immutability (information added to the blockchain can not be modified or removed; and (iii) resilience (each participant of the system keeps a complete copy of the blockchain, and no single point of failure can affect the availability of the stored information). Moreover, there are four major operations for a distributed ledger: (i) submitting a record to the ledger; (ii) querying the ledger; (iii) checking the validity of a record/block; and (iv) confirming a block.

3. DISTRIBUTED LEDGER BASED ACCESS CONTROL

In this section, we give the detailed design of DL-BAC that can be applied to different web applications and discuss its security features and limitations.

3.1 Overview of DL-BAC

At the core of DL-BAC is a distributed ledger that is responsible for access control policy storage, management, and

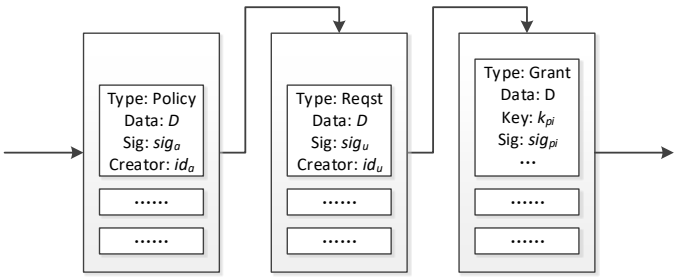

Figure 2: Three types of records stored on the distributed ledger. Multiple records can be packed into a single block and they do not need to be the same type.

enforcement. Fig. 1 gives an overview of DL-BAC. Web application data that needs protection is encrypted and stored on web servers as usual while the administrator constructs corresponding access control policies and submits them to the distributed ledger. The administrator also determines how participants involved in the distributed ledger make decisions (e.g., a certain portion of selected nodes agree to grant or deny the access request). Even if an attacker compromises the web application server or a small percentage of participants in the distributed ledger, an unauthorized user will not be able to access data under protection.

In the following description, we assume the access control policies are in the form of access control lists (ACL [31]). DL-BAC can also support more complex access control policies, which we discuss later in this section.

3.2 Major Operations of DL-BAC

DL-BAC involves three major operations for data D: (i) the administrator id_a establishes access control policy P for D; (ii) a user id_u requests access to D; and (iii) DL-BAC makes a decision to grant access to user id_u. Fig. 2 shows an example piece of the ledger used by DL-BAC that contains three types of records: access policy, data access request, and data access granting.

3.2.1 Access Control Operations

In the following, we describe four major operations in DL-BAC on access control.

Establishing access control policy. For data D, the administrator id_a prepares a list of user identities (denoted as acl_D) that have access to D, and generates a digital signature of acl to protect its integrity/authenticity. Both acl and the corresponding signature are sent to the distributed ledger and stored in the system as a block. To prevent an attacker from learning D by compromising the web server, the administrator encrypts D before sending to the web server using AES with key dek. Instead of sharing dek with any single party, the administrator uses a secret sharing protocol [34] to divide dek into multiple pieces and distribute to different participants of the distributed ledger. Specifically, id_a selects n nodes from the system of distributed ledger and sets a threshold k, divides dek to n sub-keys $dek_1, \ldots, dek_n$ according to the selected secret sharing scheme. id_a then encrypts dek_i with the ith node public key and submits the cipher-text to the distributed ledger. The secret sharing

scheme guarantees that when k or more selected nodes disclose their dek_is to a user, the user can re-construct dek.

Requesting access. When the user id_u visits the web application and needs access to data D, the application queries the distributed ledger for acl of D. If $id_u \in acl$, the web application shares D with id_u (in cipher-text) and directs the user to the distributed ledger to request access. Specifically, the user submits the request $(D, id_u, sig_u(D, id_u))$, participants of the distributed ledger verify the request, embed the request into a block, and put on the distributed ledger.

Making decisions and granting access. Each participant id_p monitors the distributed ledger for data access requests. For a request from id_u for data D accepted by the ledger, id_p first checks whether he/she is selected to protect D. If id_p is selected, he/she searches the corresponding acl for id_u to see whether id_u is authorized to access D. If $id_u \in acl$, id_p encrypts his/her piece of encryption key for D with id_u's public key and submits to the distributed ledger. After the number of key pieces submitted to the system reaches the threshold set by the data owner, a new block (grant block) is created to store collected pieces of the key. id_u can recover the original key used to encrypted D and decrypt received cipher-text.

Modifying an access policy. To add a new user to an existing access control list, the administrator can create a new acl and link it to the previous one. Because distributed ledger does not support deleting/changing of existing records, user revocation is also implemented by adding a new record to the distributed ledger states that the privilege of a certain user (or a set of users) is revoked. Changing privilege of a user is equivalent to revoking and adding operations. If a revoked user has obtained the access privilege, i.e., a grant block is created and added to the blockchain, the data owner also needs to use to new key to encrypt the data and remove the old version.

Note that if a more complex access policy framework is required (e.g., RBAC [30]), the administrator can store corresponding access policies to the ledger, the user submits his/her role information and distributed ledger participants follow saved policies to make a decision on the access request. Other access control framework like ABAC [16] can be implemented similarly.

3.2.2 *Local Data Arrangement for Distributed Ledger Participants*

Although all records are logically linked in a linear structure, each participant of the distributed ledger can organize information in a more convenient way, i.e., using a database system to store the records to quickly determine whether an access request is legitimate or not. Introducing a more complex storage structure does not affect features and operations of distributed ledger as a participant can easily track the order of all blocks on his/her local machine to determine whether a new block should be accepted and how to produce a new block (determine the prior block to start mining).

3.3 Security Analysis

Security of DL-BAC relies on features provided by the distributed ledger and underlying secret sharing scheme.

Integrity of access control policy. Because acls are stored in the distributed ledger, no one can modify them due to the immutability feature of the distributed ledger. An attacker cannot compromise the integrity of an acl by adding a patch to the distributed ledger as the attacker has to forge a digital signature of the administrator who owns the data under protection. Otherwise participants of the distributed ledger will reject the request of adding the patch. Therefore, as long as the administrator does not disclose his/her private key for signature generation, an attacker cannot tamper the integrity of access control policies by forming a valid modification request.

Confidentiality of data under protection. Data under protection is stored in encrypted form on the web server and an attacker cannot learn any useful information by compromising the web server. Although the divided data encryption key pieces are stored on the distributed ledger and publicly available, they are encrypted with different public keys (public keys of these participants selected by the administrator). Even if the attacker compromises some of the participants and gets their private keys, the secret sharing scheme guarantees that he/she cannot recover the data encryption key if the number of compromised participants does not reach the threshold. Another type of records on the distributed ledger that contain data encryption key information are *Grant* records. As long as the attacker does not know corresponding user's private key, he/she cannot recover plain-text of data encryption key.

Auditability. Auditing is an essential component of an access control system [31], and DL-BAC provides good support for auditing. The distributed ledger records all activities include construction of access control policy, access requests, and access grants. Because of the immutability of distributed ledgers, it is very hard for an attacker to alter information. Therefore, an auditor can depend on data stored in the ledger to do any auditing work.

Privacy. Although DL-BAC achieves the goal of implementing access control without any single trusted party and has the desirable security features described above, it does not provide a good protection of the users' privacy: a user's access request and granting activities are stored in plaintext on the distributed ledger that is accessible by everyone. To address the privacy concern, we give a privacy enhanced version of DL-BAC in the next section.

4. ENHANCED DL-BAC WITH PRIVACY PROTECTION

In this section, we describe the design of enhanced DL-BAC that provides privacy protection for users' activities.

4.1 Anonymized Access Request and Granting

To protect access privacy of a user, DL-BAC leverages zero-knowledge proof techniques to hide users' information [14, 9]. The basic idea is that, instead of storing the allowed users' identities to the distributed ledger, the administrator derives something from information that is only available to his/herself and allowed users as access control list acl and stores to the ledger. When a legitimate user needs to access the data, he/she generates a zero knowledge proof that shows his/her identity is in the acl without revealing it. Furthermore, this proof can be re-randomized when he/she submits a request again, so an attacker cannot learn the relationship between multiple requests by observing the distributed ledger.

Privacy enhanced DL-BAC is inspired by the anonymous membership protocol developed by Camenisch et.al. [9] which works for a two party scenario. The proposed system uses bilinear map [23] as a building block. Let $\mathbb{G}_1$ and $\mathbb{G}_T$ be two groups and $g \in \mathbb{G}_1$ is generator with prime order, a bilinear map $e : \mathbb{G}_1 \times \mathbb{G}_1 \rightarrow \mathbb{G}_T$ satisfies: (i) for $a, b \in \mathbb{Z}_p$, $e(g^a, g^b) = e(g, g)^{ab}$; and (ii) $e(g, g) \neq 1$. In practice, bilinear map can be constructed with elliptic curves and efficiently computed [19, 6, 39]. In the following description, we suppose $G_1, G_T, g \in G_1, h \in G_1$ are public parameters where g is a generator of G_1 of prime order.

Establishing access control policy. Without lose of generality, the administrator id_a numbers users in the *acl* as $u_1, \ldots, u_\ell$, and their public/private key pairs are

$$((pk_1, sk_1), \ldots, (pk_\ell, sk_\ell)).$$

User u_i also shares a secret value δ_i with the administrator id_a (e.g., δ_i can be generated by using sk_i to encrypts identity of data D). Instead of storing their identities (public keys) to the distributed ledger directly, id_a randomly selects $x \in \mathbb{Z}_p$, and publish $y \leftarrow g^x$ and $A_{\delta_i} \leftarrow g^{\frac{1}{x+\delta_i}}$ to the distributed ledger. The data encryption key is distributed in the same way as described in Section 3.

Requesting access. For user u_i to request access, he/she randomly selects five numbers v, r, s, t, m from $\mathbb{Z}_p$ and computes: $C \leftarrow g^{\delta_i} h^r, V \leftarrow A_{\delta_i}^v, a \leftarrow e(V, g)^{-s} \cdot e(g, g)^t$, and $D \leftarrow g^s h^m$. Then u_i gets another random number c by applying the hash function to the last block on the distributed ledger (i.e., $c \leftarrow$ *hash(block)%p*), and calculates $z_{\delta_i} \leftarrow s - \delta_i c, z_v \leftarrow t - vc$, and $z_r \leftarrow m - rc$. Note that c is a random number that u_i cannot control, which is essential for the security of the system.

To protect the identity, u_i cannot include his/her public key in the access request. Instead, u_i randomly generates a new public/private key pair (pk_i', sk_i') for access of data D. Let $Rqst \leftarrow (V, a, D, z_{\delta_i}, z_v, z_r)$, u_i generates the request

$$(D, Rqst, pk_i', sig_{sk_i'}(D, Rqst, pk_i'))$$

and submits to the distributed ledger.

Making decision and granting access. When a related participant id_p of the distributed ledger detects a new block including the access request is confirmed, he/she first checks whether the request is valid (e.g., whether he/she is responsible for D and the signature is valid). If the request is valid for id_p, he/she checks

- $D \stackrel{?}{=} C^c h^{z_r} g^{z_\delta}$; and
- $a \stackrel{?}{=} e(V, y)^c \cdot e(V, g)^{-z_\delta} \cdot e(g, g)^{z_v}$.

If the request passes all two checks, id_p shares the data encryption key in the same way as the original version of DL-BAC with pk_i'.

Modifying access policy. An access policy is revised by the administrator similarly to the basic DL-BAC. The major difference is that in order to add a new user to an existing list, the administrator needs to interact with the new user to obtain corresponding δ value before generating the patch policy.

Unlike the basic version of DL-BAC, privacy enhanced DL-BAC does not support access control frameworks other than access control lists. The reason for this limitation is that the zero-knowledge proof protocol used in enhanced DL-BAC only supports membership proof. We plan to develop more powerful zero-knowledge proof schemes to support other access control frameworks in the future.

4.2 Security Evaluation of Enhanced DL-BAC

Security of the anonymous user verification. Enhanced DL-BAC utilizes a zero-knowledge proof scheme to allow distributed ledger participants to determine whether a user is authorized to access certain data. A zero-knowledge proof satisfies three security properties: *completeness*, *soundness*, and *zero-knowledge* [7, 14]. Therefore, enhanced DL-BAC guarantees that only authorized users can pass the verification (completeness and soundness) and the distributed ledger cannot learn any information about the identity of the user (zero-knowledge). If the same user submits access request multiple times, he/she can re-randomize the request by choosing different random values (e.g., v, r, s, t and m) and distributed leger participants cannot link these requests. We refer the readers to [9] for more detailed formal proofs of features of the zero-knowledge proof scheme.

Policy integrity and data confidentiality. The integrity of access control policies is preserved because enhanced DL-BAC also utilizes distributed ledger to store related information. Data confidentiality is protected by the zero-knowledge proof and secret sharing scheme together. Specifically, the zero-knowledge proof protocol guarantees that only users included in the access control policy can pass the verification. The secret sharing scheme prevents unauthorized users from learning the data encryption key.

Auditability. With enhanced DL-BAC, one with access to the distributed ledger cannot do any auditing job freely because of the zero-knowledge proof protocol used to verify access requests. However, the administrator has extra knowledge of δ_i which allows him/her to detect user identities by observing the distributed ledger and conduct an audit. Specifically, given an request $Rqst \leftarrow (V, a, D, z_\delta, z_v, z_r)$ where $z_\delta \leftarrow s - \delta c, z_v \leftarrow t - vc$, and $z_r \leftarrow m - rc$, the administrator knows $\Delta = \{\delta_1, \ldots, \delta_\ell\}$ and c (generated from a block on the ledger), and for each $\delta_i \in \Delta$, he/she does following calculations:

1. $s' \leftarrow z_\delta + \delta_i c$;
2. $y \leftarrow D/g^{s'} = g^{s-s'} h^m$;
3. $x \leftarrow h^{m-rc} \cdot y^{-1} \cdot (g^\delta h^r)^c = g^{s'-s} g^{c\delta}$;
4. Checking whether $x \stackrel{?}{=} g^{c\delta}$.

If the above equation holds, the administrator knows that $\delta = \delta_i$ and recovers the identity information. In summary, with extra computations, the administrator has the capability to audit all activities related to data managed by him/her.

5. PERFORMANCE EVALUATION OF DL-BAC

In this section, we first evaluate the performance of the basic version of DL-BAC and then discuss the enhanced DL-BAC.

5.1 Performance of Basic DL-BAC

Computation and storage cost. If the underlying distributed ledger is constructed using proof-of-work [24, 18], then major computation cost of basic DL-BAC is divided into two parts, block construction and secret sharing operations. Even for early days secret sharing schemes like the one

developed by Shamir [34], only simple polynomial operations are required and it is negligible for modern computers.

From the storage perspective, DL-BAC is not as efficient as centralized access control systems because each participant has to keep a copy of all activity history. This can be mitigated with similar strategies of Bitcoin where most participants just keep most recent blocks and there are only limited number of full nodes that hold a full activity history [13].

Latency and throughput. After the administrator sets the access policy, two blocks have to be added to the distributed ledger before a user can access the data. As basic DL-BAC only adds limited computation burden, the latency is mainly determined by the underlying distributed ledger construction protocol. Considering the nature of access control, it cannot tolerate high latency. As the number of users increases, the throughput also has to be improved. There are a lot of works focusing on distributed ledger performance improvement that DL-BAC can leverage to achieve latency at second level and thousands of transactions per minute [11, 10, 21].

5.2 Performance of Enhanced DL-BAC

Compared with the basic DL-BAC, the enhanced DL-BAC requires extra computations related to the zero-knowledge proof scheme. From the administrator perspective, he/she needs to do a number of finite field inversions and elliptic curve scalar multiplications according to the length of the access control list. These operations usually can be done in milliseconds on modern computers. For each access request, the user only needs to do two bilinear pairings and several elliptic curve scalar multiplications. Distributed ledger participants have similar computation burden. All these operations are cheap and have very limited effects on the system latency and throughput. The storage cost of enhanced DL-BAC is similar to the basic version.

6. RELATED WORKS

In this section, we shortly review related works.

Distributed access control system. Researchers have developed different types of distributed access control systems [27, 3, 12]. These access control mechanisms are distributed and usually more reliable and suitable for various web applications. However, these schemes still require some trusted parties to guarantee the enforcement of access policies, which is different from DL-BAC. Zyskind and Nathan proposed the use of blockchain for personal data protection [41], which also involves data access control. However, it did not give a concrete access control approach and did not consider privacy issues of the access procedure itself.

Privacy protection in access control. Privacy protection research in the scenario of access control focuses on privacy preserving authentication [28, 25, 33, 20]. This line of research is orthogonal with our work and some results may be adoptable to the DL-BAC framework. There are also some works on transaction anonymization on blockchain [22, 32]. These works also utilize zero-knowledge technique but cannot be applied to our case directly as they do not allow multiple-usages of proof to prevent double-spending.

7. CONCLUSION

This paper proposes DL-BAC, a distributed ledger based access control system for different web applications. DL-BAC uniquely integrates cryptography tools and distributed ledger so that the enforcement of access control policies is guaranteed without any single trusted party. The enhanced DL-BAC further addresses potential privacy concerns where only the administrator who deploys access policies can audit related activities. We also evaluate the security features and performance of DL-BAC.

For the next step, we plan to implement a fully functional prototype of DL-BAC and evaluate it with different system sizes and applications. We will also extend the privacy enhanced DL-BAC to support more complex access policies including role-based access control and attribute-based access control.

8. REFERENCES

[1] FIPS PUB 197: Advanced Encryption Standard, November 2001.

[2] Carlisle Adams and Steve Lloyd. *Understanding PKI: Concepts, Standards, and Deployment Considerations*. Addison-Wesley Professional, 2003.

[3] Abdulrahman Almutairi, Muhammad Sarfraz, Saleh Basalamah, Walid Aref, and Arif Ghafoor. A distributed access control architecture for cloud computing. *IEEE software*, 29(2):36, 2012.

[4] Elisa Bertino, Claudio Bettini, Elena Ferrari, and Pierangela Samarati. A temporal access control mechanism for database systems. *IEEE Transactions on knowledge and data engineering*, 8(1):67–80, 1996.

[5] Elisa Bertino, Sushil Jajodia, and Pierangela Samarati. Supporting multiple access control policies in database systems. In *Security and Privacy, 1996. Proceedings., 1996 IEEE Symposium on*, pages 94–107. IEEE, 1996.

[6] Ian F. Blake, V. Kumar Murty, and Guangwu Xu. Refinements of Miller's algorithm for computing the weil/tate pairing. *Journal of Algorithms*, 58:134–149, February 2006.

[7] Manuel Blum, Paul Feldman, and Silvio Micali. Non-interactive zero-knowledge and its applications. In *Proceedings of the twentieth annual ACM symposium on Theory of computing*, pages 103–112. ACM, 1988.

[8] Vitalik Buterin. What proof of stake is and why it matters. *Bitcoin Magazine, August*, 26, 2013.

[9] Jan Camenisch, Rafik Chaabouni, et al. Efficient protocols for set membership and range proofs. In *International Conference on the Theory and Application of Cryptology and Information Security*, pages 234–252. Springer, 2008.

[10] Nicolas T Courtois, Pinar Emirdag, and Daniel A Nagy. Could bitcoin transactions be 100x faster? In *Security and Cryptography (SECRYPT), 2014 11th International Conference on*, pages 1–6. IEEE, 2014.

[11] Ittay Eyal, Adem Efe Gencer, Emin Gün Sirer, and Robbert Van Renesse. Bitcoin-ng: A scalable blockchain protocol. In *13th USENIX Symposium on Networked Systems Design and Implementation (NSDI 16)*, pages 45–59, 2016.

[12] Eric Freudenthal, Tracy Pesin, Lawrence Port, Edward Keenan, and Vijay Karamcheti. dRBAC:

distributed role-based access control for dynamic coalition environments. In *Distributed Computing Systems, 2002. Proceedings. 22nd International Conference on*, pages 411–420. IEEE, 2002.

[13] Juan Garay, Aggelos Kiayias, and Nikos Leonardos. The Bitcoin backbone protocol: Analysis and applications. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pages 281–310. Springer, 2015.

[14] Oded Goldreich, Silvio Micali, and Avi Wigderson. Proofs that yield nothing but their validity or all languages in NP have zero-knowledge proof systems. *Journal of the ACM (JACM)*, 38(3):690–728, 1991.

[15] Nozomi Hayase. The Blockchain and the Rise of Networked Trust. `http://www.coindesk.com/blockchain-rise-networked-trust/`, 2014.

[16] Vincent C Hu, D Richard Kuhn, and David F Ferraiolo. Attribute-based access control. *Computer*, (2):85–88, 2015.

[17] James BD Joshi, Walid G Aref, Arif Ghafoor, and Eugene H Spafford. Security models for web-based applications. *Communications of the ACM*, 44(2):38–44, 2001.

[18] Sunny King. Primecoin: Cryptocurrency with prime number proof-of-work. 2013.

[19] Peter L. Montgomery Kirsten Eisenträger, Kristin Lauter. Fast elliptic curve arithmetic and improved Weil pairing evaluation. In *Topics in Cryptology - CT-RSA 2003*, volume 2612 of *LNCS*, 2003.

[20] Andrew Yehuda Lindell. Anonymous authentication. *Journal of Privacy and Confidentiality*, 2(2):4, 2007.

[21] Loi Luu, Viswesh Narayanan, Kunal Baweja, Chaodong Zheng, Seth Gilbert, and Prateek Saxena. SCP: a computationally-scalable byzantine consensus protocol for blockchains. Technical report, Cryptology ePrint Archive, Report 2015/1168, 2015.

[22] Ian Miers, Christina Garman, Matthew Green, and Aviel D Rubin. Zerocoin: Anonymous distributed e-cash from bitcoin. In *Security and Privacy (SP), 2013 IEEE Symposium on*, pages 397–411. IEEE, 2013.

[23] Victor S. Miller. The Weil pairing, and its efficient calculation. *Journal of Cryptology*, 17(4):235–261, September 2004.

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

[25] Lan Nguyen and Rei Safavi-Naini. Dynamic k-times anonymous authentication. In John Ioannidis, Angelos Keromytis, and Moti Yung, editors, *Applied Cryptography and Network Security - ACNS 2005*, volume 3531 of *LNCS*, pages 318–333. Springer, 2005.

[26] Marc Pilkington. Blockchain technology: principles and applications. *Research Handbook on Digital Transformations, edited by F. Xavier Olleros and Majlinda Zhegu. Edward Elgar*, 2016.

[27] Sushmita Ruj, Amiya Nayak, and Ivan Stojmenovic. DACC: Distributed access control in clouds. In *2011IEEE 10th International Conference on Trust, Security and Privacy in Computing and Communications*, pages 91–98. IEEE, 2011.

[28] Sushmita Ruj, Milos Stojmenovic, and Amiya Nayak. Privacy preserving access control with authentication for securing data in clouds. In *Cluster, Cloud and Grid Computing (CCGrid), 2012 12th IEEE/ACM International Symposium on*, pages 556–563. IEEE, 2012.

[29] Pierangela Samarati and Sabrina Capitani de Vimercati. Access control: Policies, models, and mechanisms. In *International School on Foundations of Security Analysis and Design*, pages 137–196. Springer, 2000.

[30] Ravi S Sandhu, Edward J Coyne, Hal L Feinstein, and Charles E Youman. Role-based access control models. *Computer*, 29(2):38–47, 1996.

[31] Ravi S Sandhu and Pierangela Samarati. Access control: principle and practice. *Communications Magazine, IEEE*, 32(9):40–48, 1994.

[32] Eli Ben Sasson, Alessandro Chiesa, Christina Garman, Matthew Green, Ian Miers, Eran Tromer, and Madars Virza. Zerocash: Decentralized anonymous payments from bitcoin. In *2014 IEEE Symposium on Security and Privacy*, pages 459–474. IEEE, 2014.

[33] Stuart Schechter, Todd Parnell, and Alexander Hartemink. Anonymous authentication of membership in dynamic groups. In *International Conference on Financial Cryptography*, pages 184–195. Springer, 1999.

[34] Adi Shamir. How to share a secret. *Communications of the ACM*, 22:612–613, 1979.

[35] Melanie Swan. *Blockchain: Blueprint for a new economy.* " O'Reilly Media, Inc.", 2015.

[36] Tim Thomas. A mandatory access control mechanism for the unix file system. In *Aerospace Computer Security Applications Conference, 1988., Fourth*, pages 173–177. IEEE, 1988.

[37] Sarah Underwood. Blockchain beyond bitcoin. *Communications of the ACM*, 59(11):15–17, 2016.

[38] Marko Vukolić. The quest for scalable blockchain fabric: Proof-of-work vs. BFT replication. In *International Workshop on Open Problems in Network Security*, pages 112–125. Springer, 2015.

[39] Lei Xu and Dongdai Lin. Refinement of Miller's algorithm over Edwards curves. In Josef Pieprzyk, editor, *Topics in Cryptology - CT-RSA 2010*, volume 5985 of *LNCS*, pages 106–118. Springer, 2010.

[40] Shucheng Yu, Cong Wang, Kui Ren, and Wenjing Lou. Achieving secure, scalable, and fine-grained data access control in cloud computing. In *Proceedings of the 29th conference on Information communications - INFOCOM 2010*, pages 534–542. IEEE Press, 2010.

[41] Guy Zyskind, Oz Nathan, et al. Decentralizing privacy: Using blockchain to protect personal data. In *Security and Privacy Workshops (SPW), 2015 IEEE*, pages 180–184. IEEE, 2015.