Review of Decentralizing Privacy: Using Blockchain to Protect Personal Data

Brian Ly
York University
4700 Keele St, Toronto, ON M3J 1P3
Lybrian1@my.yorku.ca

Simin Lin
York University
4700 Keele St, Toronto, ON M3J 1P3
Simon001@my.yorku.ca

Abstract

In recent years there have been an increase in surveillance and security breaches. Crypto currencies such as Bitcoin has a reputation for being trusted by using a decentralized network of peers accompanied by a public ledger. This paper describes a method of implementing a protocol that turns a blockchain into an automated access-control manager that does not require trust in a third party like current methods. Instead, this method utilizes blockchain technology to carry instructions such as storing, querying and sharing data.

1. Introduction

In this current Big Data era, the amount of data generated is increasing dramatically. In fact, a full 90% of all the data in the world has been generated just over the last two years [2]. Data is constantly being collected and analyzed, increasing the need to secure that data. Data is becoming one of the most valuable asset in our economy, innovation and economic growth. Because of this there is a growing public concern about user privacy since an increasing amount of data is becoming available on the internet. Each and every one of us is constantly producing and releasing data about ourselves on a daily basis [2].

2. Background

2.1. Blockchain & Bitcoin

Blockchain is a peer-to-peer digital ledger of transactions that are distributed to all users, which makes it decentralized and distributed. Blockchain uses cryptography and a consensus mechanism to verify transactions. This verification legitimizes transactions and prevents inconsistencies such as duplications. This is the reason why blockchain technology is the backbone of Bitcoin and other cryptocurrencies since it allows for high-value transactions in a trustless environment. A blockchain offers complete transparency to eliminate the need for intermediaries or third-party administrators to verify transactions.

Figure 1 below shows the basic building blocks of blockchain. Blockchain consists of numerous nodes and transactions among the nodes. Each node is an Internet-enabled device connected to the blockchain. Each node can broadcast transactions. If a transaction is valid, other nodes will accept it and uses most difficult proof-of-concept and add it to the existing honest chain. If honest nodes have more computing power than attacking nodes, the honest chain will grow faster and be recognized by all the nodes as valid.

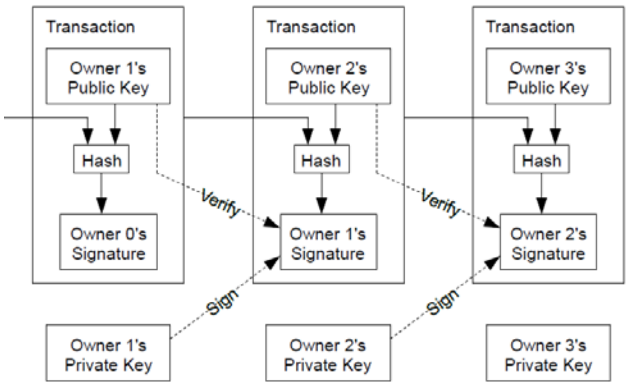

Figure 1: Diagram of Transactions in a Blockchain [6].

As one of the first implementations of blockchain, Bitcoin allows users to transfer virtual currency securely without the need of a centralized or third party regulator by using an open ledger that can be verified publically. This concept is known as a blockchain. Since then, other cryptocurrencies and projects have surfaced that used blockchain technology and demonstrated the multiple security functions blockchain can serve.

2.2. The Problem

In essence, blockchain is a structure that allows any types of information transfer. With the ever-growing concerns about Internet privacy, it is reasonable to suggest the use of blockchain in collecting mobile app information.

Privacy has long been considered a large concern in the world of Internet. Among all concerns, breach of mobile security is viral and largely uncontainable. This can be traced back to lack of trust between mobile users and third-party services. The users rely on these services for daily tasks and convenience. Unfortunately, the services usually utilize hidden cookies to gather private information about the users. For example, the premise of a fitness app is developed to keep track of users' daily exercise and maybe recommending various exercise routines best for individual users. However, these apps also audit the users' private information such weight, height, GPS location. This review paper does not focus on determining whether that information is used to benefit the users or to exploit their trust. Rather, it explores the worst case and how blockchain architecture can be used to prevent third-party services from abusing the trust.

3. Proposed Solutions

By combining blockchain and off-blockchain storage we can develop a personal data management platform that focuses on security. The proposed solution will protect against, data ownership, data transparency and auditability as well as fine-grained access control.

To demonstrate the proposed solution, we first introduce two variables, user (U) and service (S). U is an actor outside the proposed blockchain system. He wishes to have his private information secured. S refers to a mobile service that U uses. To use the proposed system, S must integrate the network operation of the blockchain into their products. This can be easily achieved by a standard software development kit. Once integrated, both U and S will have access to the blockchain, which is a disinterested network of nodes.

As shown in Figure 2, the system consists of two types of transactions, T_{access} and T_{data}. T_{access} is used for access control management while T_{data} is used for data storage and retrieval. The system accepts these two forms of transactions into the blockchain. A typical use case would be, the user installs an application on their smartphone that uses the proposed network protocol. When the user signs up for the service, a new shared identity is generated and sent to the blockchain in a T_{access} transaction type along with the associated permissions of that user. The private data collected on the phone is encrypted using a shared encryption key before being sent out to the blockchain in a T_{data} transaction. It is then routed into a off-blockchain key-value store. Only the pointer to the data, which is the SHA-256 hash of the data, is stored on the public ledger. Now both the user and the service can query the data using the T_{data} transaction. The user can change the permissions granted to the service at any time by using the T_{access} transaction with new permissions or even completely revoking access of the service. This gives the users complete control over their personal data.

In Figure 2, the distributed hash table (DHT) represents the off-blockchain key-value store which is an implementation of Kademlia, a peer-to-peer (P2P) DHT. Kademlia uses a novel XOR metric for distance between points in the key space [5]. It is the first peer-to-peer system that combines provable consistency and performance, latency-minimizing and routing in a symmetric, unidirectional topology. The DHT is maintained by a network of nodes that fulfill transactions. Data is stored randomly across the nodes and replicated many times to ensure availability. Another alternative option to improve scalability and ease of deployment would be to store the data on a centralized cloud service, but that involves some level of trust.

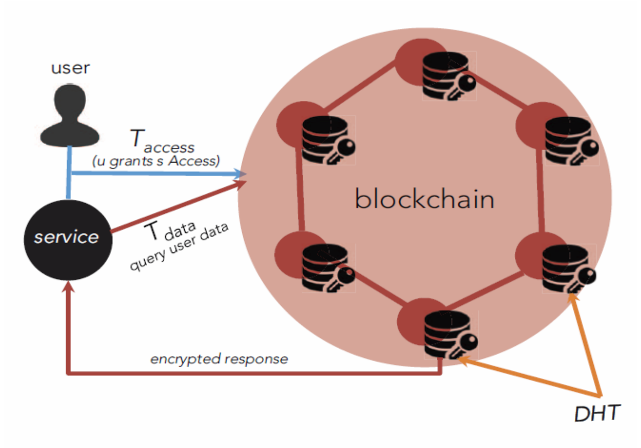

Figure 2: Diagram of Proposed Solution [1].

4. Methodology

Standard cryptographic building blocks are utilized in the platform, which is a symmetric encryption scheme and a digital signature scheme. The blockchain network protocol is implemented using the Elliptic Curve Digital Signature Algorithm (ECDSA) [3]. ECDSA is a digital signature algorithm designed to ensure a digital signature can be verified. More specifically, given a public key, the algorithm can detect if the digital signature is generated from the private key, hence, ensuring that no unauthorized users can access or query the information hidden in the blockchain.

4.1. Protocols & Implementations

There are four protocols used in building the blockchain network. Protocol 1 introduces compound identities. Under the protocol, U and S can be combined to form identity. This identity is used for both parties to access information. To protect U's autonomy, owner privilege is granted to U; whereas S is given the guest permission.

On the owner's side, the signature generator generates a public key (pk) and a private key (sk). U hides the private key while sharing the public key with S. Another private key with encrypted message is generated and is also shared with the guest. On the guest side, a set of public signature and private signature keys are generated. The public key is shared with the owner while the guest holds the private key.

Protocol 1 Generating a compound identity

```
procedure COMPOUNDIDENTITY($(u, s)$)
    $u$ and $s$ form a secure channel
    $u$ executes:
        $(pk^{u,s}_{sig}, sk^{u,s}_{sig}) \leftarrow \mathcal{G}_{sig}()$
        $sk^{u,s}_{enc} \leftarrow \mathcal{G}_{enc}()$
        $u$ shares $sk^{u,s}_{enc}, pk^{u,s}_{sig}$ with $s$
    $s$ executes:
        $(pk^{s,u}_{sig}, sk^{s,u}_{sig}) \leftarrow \mathcal{G}_{sig}()$
        $s$ shares $pk^{s,u}_{sig}$ with $s$
    // Both $u$ and $s$ have $sk^{u,s}_{enc}, pk^{u,s}_{sig}, pk^{s,u}_{sig}$
    return $pk^{u,s}_{sig}, pk^{s,u}_{sig}, sk^{u,s}_{enc}$
end procedure
```

Figure 3: Proposed System Protocol 1 [1].

The second protocol deals with checking for permissions. $POLICY_{u,s}$ is a set of information that U shares with S.

$POLICY_{u,s}$ = {*shared information*}

Since U is the owner, he always has permission to all own information. However, the same cannot be said for S. To check if S has certain permission, CHECKPOLICY(pk, x_p)is used. This method checks for two conditions. 1) The public key in the parameter is the same as the one U shares with S. 2) The public key in the parameter is the same as the one S shares with U and x_p is in already in $POLICY_{u,s}$, If either condition is met, that means S has the permission.

Protocol 2 Permissions check against the blockchain

```
procedure CHECKPOLICY($pk^k_{sig}, x_p$)
    $s \leftarrow 0$
    $a_{policy} = \mathcal{H}(pk^k_{sig})$
    if $L[a_{policy}] \neq \emptyset$ then
        $pk^{u,s}_{sig}, pk^{s,u}_{sig}, POLICY_{u,s} \leftarrow Parse(L[a_{policy}])$
        if $pk^k_{sig} = pk^{u,s}_{sig}$ or
(\$pk^k_{sig} = pk^{s,u}_{sig}$ and $x_p \in POLICY_{u,s}$) then
            $s \leftarrow 1$
        end if
    end if
    return $s$
end procedure
```

Figure 4: Proposed System Protocol 2 [1].

The third protocol is associated with T_{access} transactions and is used for access control. This protocol executes when a transaction is received. If S wants to be granted access to private information, he must invoke HANDLEACCESSTX(pk, m), where m is the message containing $POLICY_{u,s}$ and the shared public key from U to S. If the public key in the parameter matches the pubic key in m, then S is granted access to that private information.

Protocol 3 Access Control Protocol

```
procedure HANDLEACCESSTX($pk^k_{sig}, m$)
    $s \leftarrow 0$
    $pk^{u,s}_{sig}, pk^{s,u}_{sig}, POLICY_{u,s} = Parse(m)$
    if $pk^k_{sig} = pk^{u,s}_{sig}$ then
        $L[\mathcal{H}(pk^k_{sig})] = m$
        $s \leftarrow 1$
    end if
    return $s$
end procedure
```

Figure 5: Proposed System Protocol 3 [1].

Last but not least, protocol 4 deals with T_{data} transactions and data storage and retrieval. While U always has access to read and write his data, S is granted access conditionally. In HANDLEACCESSTX(pk, m), parameter m is parsed into 3 fields, *c*,*x_p* and *rw*. where c is a string, $x_p \in POLICY_{u,s}$ and rw ∈ {0,1}. Protocol 2 is used to determine if S has permission to access certain information. If he does, the procedure then checks for the types of permission. If S has write permission, c is hashed through a cryptographic hash function and added to the blockchain memory space. If S only has read permission and c is in the blockchain memory space, then return c being hashed to a distributed hash table.

Protocol 4 Storing or Loading Data

```
procedure HANDLEDATATX($pk^k_{sig}, m$)
    $c, x_p, rw = Parse(m)$
    if $CheckPolicy(pk^k_{sig}, x_p) = \texttt{True}$ then
        $pk^{u,s}_{sig}, pk^{s,u}_{sig}, POLICY_{u,s} \leftarrow$
$Parse(L[\mathcal{H}(pk^{u,s}_{sig})])$
        $a_{x_p} = \mathcal{H}(pk^{u,s}_{sig} \parallel x_p)$
        if $rw = 0$ then          ▷ rw=0 for write, 1 for read
            $h_c = \mathcal{H}(c)$
            $L[a_{x_p}] \leftarrow L[a_{x_p}] \cup h_c$
            (DHT) $ds[h_c] \leftarrow c$
            return $h_c$
        else if $c \in L[a_{x_p}]$ then
            (DHT) return $ds[h_c]$
        end if
    end if
    return $\emptyset$
end procedure
```

Figure 6: Proposed System Protocol 4 [1].

4.2. Additional Information

While discussing protocol 4, the cryptographic hash function was mentioned. A cryptographic hash function (Figure 7) is a hash function that takes an input string and returns a fixed-size alphanumeric string. Assuming the function is instantiated by a SHA-256 implementation, the algorithm will return a 256-bit message [8].

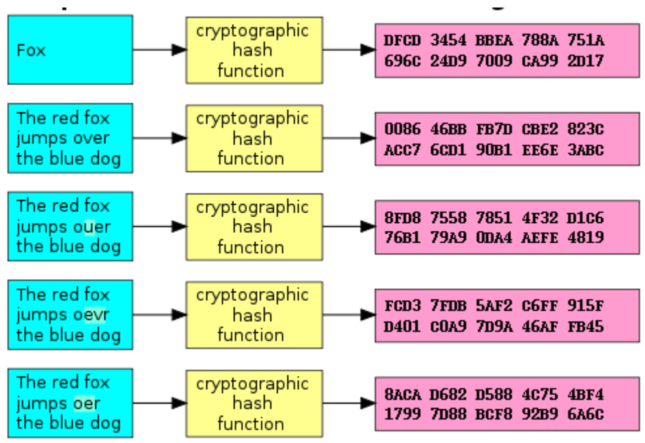
Fox
The red fox jumps over the blue dog
The red fox jumps ouer the blue dog
The red fox jumps oevr the blue dog
The red fox jumps oer the blue dog
cryptographic hash function
DFCD 3454 BBEA 788A 751A 696C 24D9 7009 CA99 2017
0086 46BB FB7D CBE2 823C ACC7 6CD1 90B1 EE6E 3ABC
8FD8 7558 7851 4F32 D1C6 76B1 79A9 0DA4 AEFE 4819
FCD3 7FDB 5AF2 C6FF 915F D401 C0A9 7D9A 46AF FB45
8ACA D682 D588 4C75 4BF4 1799 7D88 BCF8 92B9 6A6C

Figure 7: Cryptographic Hash Function.

The function outputs the result in 8 hashed blocks, each of which is 32-bit.

$$H_0^{(N)} | H_1^{(N)} | H_2^{(N)} | H_3^{(N)} | H_4^{(N)} | H_5^{(N)} | H_6^{(N)} | H_7^{(N)}$$

$$H_i^{(j)} = hashed\ message + H_i^{(j-1)} \qquad for\ i = 0...7;\ for\ j = 0...7$$

$$H_0^{(0)} = 6a09e667$$
$$H_1^{(0)} = bb67ae85$$
$$H_2^{(0)} = 3c6ef372$$
$$H_3^{(0)} = a54ff53a$$
$$H_4^{(0)} = 510e527f$$
$$H_5^{(0)} = 9b05688c$$
$$H_6^{(0)} = 1f83d9ab$$
$$H_7^{(0)} = 5be0cd19$$

5. Future Extension

One of the major hurdles is to overcome the public nature of the blockchain. While the data is encrypted, the service can query a piece of raw data and store it for future analysis. The proposed solution is to not give the service any access to the raw data. Instead, the service can run computations directly on the network and only receive the final results. Furthermore, the data can be split into shares instead of encrypting them. Secure Multi-party Computation (MPC) which is one of the central concepts in modern cryptography can be used to securely evaluate any functions it might have [4]. In MPC, the network selects a subset of nodes at random to commute a secure version of the data and an interpreter transforms the code into a MPC protocol. The results are broadcasts back to the entire network and stored on a public ledger.

The principle of blockchain is that it assumes all nodes are equally untrustworthy [6]. Their decision-making process is collective, based on the Proof-of-work algorithm. In Proof-of-work, it makes decisions basically with one-CPU-one-vote. The majority decision is represented by the longest blockchain which also has the greatest proof-of-work effort invested in it [6]. Assuming that the longest chain is controlled by honest nodes which make up the majority of CPU power, the honest chain will keep growing and outpacing any dishonest chains. Essentially, to attack the blockchain, the attacker would have to recompute all the proof-of-work and surpass all the honest nodes [6]. Proof-of-Work assumes that the nodes that contribute significant resources into the blockchain are less likely to cheat and that honest actors are rewarded.

This may lead to a vulnerability to sybil attacks, where attackers subvert the reputation system of a peer-to-peer network by creating a large number of nodes and using them to gain a disproportionately large influence over the decision making [7]. A simple solution to this would be to count the number of good and bad actions each node takes. Using the sigmoid function to evaluate its trust score, the network can give more weight to the nodes that score higher and compute the blocks more efficiently.

$$trust_n^{(i)} = \frac{1}{1 + e^{-\alpha(\#good - \#bad)}},$$

where α is the step size. This mechanism should be resistant to sybil attacks since it takes time to earn trust in the system. Although an attacker can still potentially build up their trust score just to maliciously attack at a later time, this can be mitigated by simply randomly selecting several nodes to vote on each block then take their equally weighted majority vote.

6. Conclusion

Sensitive data should not be trusted to intermediaries or third-parties because they are susceptible to attacks and misuse. Data owners should control their own data without compromising security and be able to decide who has access to their data. The proposed platform combines a blockchain, as an access control moderator with an off-blockchain storage solution. This solution does not require the user to trust intermediaries or third-parties and can focus on utilizing data without being concerned about security. Furthermore, the user can revoke access to their data at any time and services do not have to be overly concerned about properly securing the data. Additionally, the blockchain can act as a tamper-proof ledger for legal proceedings where legal evidence for accessing or storing data is required.

References

[1] G. Zyskind, O. Nathan and A. '. Pentland, "Decentralizing Privacy: Using Blockchain to Protect Personal Data," *2015 IEEE Security and Privacy Workshops*, San Jose, CA, 2015, pp. 180-184.

[2] SINTEF. "Big Data, for better or worse: 90% of world's data generated over last two years." *ScienceDaily*. ScienceDaily, 22 May 2013. <www.sciencedaily.com/releases/2013/05/130522085217.htm>.

[3] Johnson, Don, et al. "The Elliptic Curve Digital Signature Algorithm (ECDSA)." *International Journal of Information Security*, vol. 1, no. 1, 2001, pp. 36–63., doi:10.1007/s102070100002.

[4] Ben-Or, Michael, and Avi Wigderson. "Completeness Theorems for Non-Cryptographic Fault-Tolerant Distributed Computation." *Proceedings of the Twentieth Annual ACM Symposium on Theory of Computing - STOC 88*, 1988, doi:10.1145/62212.62213.

[5] Maymounkov, Petar, and David Mazières. "Kademlia: A Peer-to-Peer Information System Based on the XOR Metric." *Peer-to-Peer Systems Lecture Notes in Computer Science*, 2002, pp. 53–65., doi:10.1007/3-540-45748-8_5.

[6] Nakamoto, Satoshi. "Bitcoin: A Peer-to-Peer Electronic Cash System." *Bitcoin*, Mar. 2009, bitcoin.org/bitcoin.pdf.

[7] Douceur J.R. (2002) The Sybil Attack. In: Druschel P., Kaashoek F., Rowstron A. (eds) Peer-to-Peer Systems. IPTPS 2002. *Lecture Notes in Computer Science*, vol 2429. Springer, Berlin, Heidelberg

[8] "FEDERAL INFORMATION PROCESSING STANDARDS Secure Hash Standard." *Federal Information*, National Institute of Standards and Technology , Aug. 2015, nvlpubs.nist.gov/nistpubs/FIPS/NIST.FIPS.180-4.pdf.