Decentralizing Privacy: Using Blockchain to Protect Personal Data

Guy Zyskind
MIT Media Lab
Cambridge, Massachusetts
Email: guyz@mit.edu

Oz Nathan
Tel-Aviv University
Tel-Aviv, Israel
Email: oznathan@gmail.com

Alex 'Sandy' Pentland
MIT Media Lab
Cambridge, Massachusetts
Email: pentland@mit.edu

***Abstract*—The recent increase in reported incidents of surveillance and security breaches compromising users' privacy call into question the current model, in which third-parties collect and control massive amounts of personal data. Bitcoin has demonstrated in the financial space that trusted, auditable computing is possible using a decentralized network of peers accompanied by a public ledger. In this paper, we describe a decentralized personal data management system that ensures users own and control their data. We implement a protocol that turns a blockchain into an automated access-control manager that does not require trust in a third party. Unlike Bitcoin, transactions in our system are not strictly financial – they are used to carry instructions, such as storing, querying and sharing data. Finally, we discuss possible future extensions to blockchains that could harness them into a well-rounded solution for trusted computing problems in society.**

***Keywords*—blockchain; privacy; bitcoin; personal data**

I. Introduction

The amount of data in our world is rapidly increasing. According to a recent report [22], it is estimated that 20% of the world's data has been collected in the past couple of years. Facebook, the largest online social-network, collected 300 petabytes of personal data since its inception [1] – a hundred times the amount the Library of Congress has collected in over 200 years [13]. In the Big Data era, data is constantly being collected and analyzed, leading to innovation and economic growth. Companies and organizations use the data they collect to personalize services, optimize the corporate decision-making process, predict future trends and more. Today, data is a valuable asset in our economy [21].

While we all reap the benefits of a data-driven society, there is a growing public concern about user privacy. Centralized organizations – both public and private, amass large quantities of personal and sensitive information. Individuals have little or no control over the data that is stored about them and how it is used. In recent years, public media has repeatedly covered controversial incidents related to privacy. Among the better known examples is the story about government surveillance [2], and Facebook's large-scale scientific experiment that was apparently conducted without explicitly informing participants [10].

Related Work. There have been various attempts to address these privacy issues, both from a legislative perspective ([4], [20]), as well as from a technological standpoint. OpenPDS, a recently developed framework, presents a model for autonomous deployment of a PDS which includes a mechanism for returning computations on the data, thus returning answers instead of the raw data itself [6]. Across the industry, leading companies chose to implement their own proprietary authentication software based on the OAuth protocol [19], in which they serve as centralized trusted authorities.

From a security perspective, researchers developed various techniques targeting privacy concerns focused on personal data. Data anonymization methods attempt to protect personally identifiable information. *k-anonymity*, a common property of anonymized datasets requires that sensitive information of each record is indistinguishable from at least $k-1$ other records [24]. Related extensions to *k-anonymity* include *l-diversity*, which ensures the sensitive data is represented by a diverse enough set of possible values [15]; and *t-closeness*, which looks at the distribution of sensitive data [14]. Recent research has demonstrated how anonymized datasets employing these techniques can be de-anonymized [18], [5], given even a small amount of data points or high dimensionality data. Other privacy-preserving methods include *differential privacy*, a technique that perturbs data or adds noise to the computational process prior to sharing the data [7], and encryption schemes that allow running computations and queries over encrypted data. Specifically, fully homomorphic encryption (FHE) [9] schemes allow any computation to run over encrypted data, but are currently too inefficient to be widely used in practice.

In recent years, a new class of accountable systems emerged. The first such system was Bitcoin, which allows users to transfer currency (bitcoins) securely without a centralized regulator, using a publicly verifiable open ledger (or *blockchain*). Since then, other projects (collectively referred to as *Bitcoin 2.0* [8]) demonstrated how these blockchains can serve other functions requiring trusted computing and auditability.

Our Contribution. 1) We combine blockchain and off-blockchain storage to construct a personal data management platform focused on privacy. 2) We illustrate through our platform and a discussion of future improvements to the technology, how blockchains could become a vital resource in trusted-computing.

Organization. Section II discusses the privacy problem we solve in this paper; section III provides an overview of the platform, whereas section IV describes in detail the technical implementation; section V discusses future extensions to blockchains, and concluding remarks are found in section VI.

The first two authors contributed equally to this work.

DOI 10.1109/SPW.2015.27

II. The Privacy Problem

Throughout this paper, we address the privacy concerns users face when using third-party services. We focus specifically on mobile platforms, where services deploy applications for users to install. These applications constantly collect high-resolution personal data of which the user has no specific knowledge or control. In our analysis, we assume that the services are *honest-but-curious* (i.e., they follow the protocol). Note that the same system could be used for other data-privacy concerns, such as patients sharing their medical data for scientific research, while having the means to monitor how it is used and the ability to instantly opt-out. In light of this, our system protects against the following common privacy issues:

Data Ownership. Our framework focuses on ensuring that users own and control their personal data. As such, the system recognizes the users as the owners of the data and the services as guests with delegated permissions.

Data Transparency and Auditability. Each user has complete transparency over what data is being collected about her and how they are accessed.

Fine-grained Access Control. One major concern with mobile applications is that users are required to grant a set of permissions upon sign-up. These permissions are granted indefinitely and the only way to alter the agreement is by opting-out. Instead, in our framework, at any given time the user may alter the set of permissions and revoke access to previously collected data. One application of this mechanism would be to improve the existing permissions dialog in mobile applications. While the user-interface is likely to remain the same, the access-control policies would be securely stored on a blockchain, where only the user is allowed to change them.

III. Proposed Solution

We begin with an overview of our system. As illustrated in Figure 1, the three entities comprising our system are mobile phone *users*, interested in downloading and using applications; *services*, the providers of such applications who require processing personal data for operational and business-related reasons (e.g., targeted ads, personalized service); and *nodes*, entities entrusted with maintaining the blockchain and a distributed private key-value data store in return for incentives. Note that while users in the system normally remain (pseudo) anonymous, we could store service profiles on the blockchain and verify their identity.

The system itself is designed as follows. The blockchain accepts two new types of transactions: T_{access}, used for access control management; and T_{data}, for data storage and retrieval. These network operations could be easily integrated into a mobile software development kit (SDK) that services can use in their development process.

To illustrate, consider the following example: a user installs an application that uses our platform for preserving her privacy. As the user signs up for the first time, a new shared (user, service) identity is generated and sent, along with the associated permissions, to the blockchain in a T_{access} transaction. Data collected on the phone (e.g., sensor data such as location) is encrypted using a shared encryption key and sent to the blockchain in a T_{data} transaction, which subsequently routes it to an off-blockchain key-value store, while retaining only a pointer to the data on the public ledger (the pointer is the SHA-256 hash of the data).

Both the service and the user can now query the data using a T_{data} transaction with the pointer (key) associated to it. The blockchain then verifies that the digital signature belongs to either the user or the service. For the service, its permissions to access the data are checked as well. Finally, the user can change the permissions granted to a service at any time by issuing a T_{access} transaction with a new set of permissions, including revoking access to previously stored data. Developing a web-based (or mobile) dashboard that allows an overview of one's data and the ability to change permissions is fairly trivial and is similar to developing centralized-wallets, such as Coinbase for Bitcoin[1].

The off-blockchain key-value store is an implementation of Kademilia [16], a distributed hashtable (or *DHT*), with added persistence using LevelDB[2] and an interface to the blockchain. The DHT is maintained by a network of nodes (possibly disjoint from the blockchain network), who fulfill approved read/write transactions. Data are sufficiently randomized across the nodes and replicated to ensure high availability. It is instructive to note that alternative off-blockchain solutions could be considered for storage. For example, a centralized cloud might be used to store the data. While this requires some amount of trust in a third-party, it has some advantages in terms of scalability and ease of deployment.

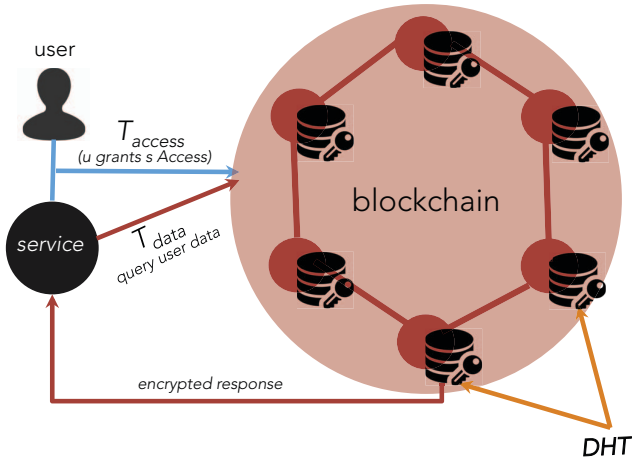

Fig. 1. Overview of the decentralized platform.

IV. The Network Protocol

We now describe in detail the underlying protocol used in the system. We utilize standard cryptographic building blocks in our platform: a symmetric encryption scheme defined by the 3-tuple $(\mathcal{G}_{enc}, \mathcal{E}_{enc}, \mathcal{D}_{enc})$ – the generator, encryption and decryption algorithms respectively; a digital signature scheme (*DSS*) described by the 3-tuple $(\mathcal{G}_{sig}, \mathcal{S}_{sig}, \mathcal{V}_{sig})$ – the generator, signature and verification algorithms respectively,

[1]Coinbase bitcoin wallet, http://www.coinbase.com
[2]LevelDB, http://github.com/google/leveldb

implemented using ECDSA with *secp256k1* curve [12]; and a cryptographic hash function $\mathcal{H}$, instantiated by a *SHA-256* [11] implementation.

A. Building Blocks

We now briefly introduce relevant building blocks that are used throughout the rest of this paper. We assume familiarity with Bitcoin [17] and blockchains.

1) Identities: Blockchains utilize a pseudo-identity mechanism. Essentially a public-key, every user can generate as many such pseudo-identities as she desires in order to increase privacy. We now introduce *compound identities*, an extension of this model used in our system. A compound identity is a shared identity for two or more parties, where some parties (at least one) own the identity (*owners*), and the rest have restricted access to it (*guests*). Protocol 1 illustrates the implementation for a single owner (the user) and a single guest (the service). As illustrated, the identity is comprised of signing key-pairs for the owner and guest, as well as a symmetric key used to encrypt (and decrypt) the data, so that the data is protected from all other players in the system. Formally, a compound identity is externally (as seen by the network) observed by the 2-tuple:

$$Compound_{u,s}^{(public)} = (pk_{sig}^{u,s}, pk_{sig}^{s,u}) \quad (1)$$

Similarly, the entire identity (including the private keys) is the following 5-tuple:

$$Compound_{u,s} = (pk_{sig}^{u,s}, sk_{sig}^{u,s}, pk_{sig}^{s,u}, sk_{sig}^{s,u}, sk_{enc}^{u,s}) \quad (2)$$

Protocol 1 Generating a compound identity

1: **procedure** COMPOUNDIDENTITY(u, s)
2: u *and* s *form a secure channel*
3: u executes:
4: $(pk_{sig}^{u,s}, sk_{sig}^{u,s}) \leftarrow \mathcal{G}_{sig}()$
5: $sk_{enc}^{u,s} \leftarrow \mathcal{G}_{enc}()$
6: u *shares* $sk_{enc}^{u,s}, pk_{sig}^{u,s}$ *with* s
7: s executes:
8: $(pk_{sig}^{s,u}, sk_{sig}^{s,u}) \leftarrow \mathcal{G}_{sig}()$
9: s *shares* $pk_{sig}^{s,u}$ *with* s
10: // Both u and s have $sk_{enc}^{u,s}, pk_{sig}^{u,s}, pk_{sig}^{s,u}$
11: **return** $pk_{sig}^{u,s}, pk_{sig}^{s,u}, sk_{enc}^{u,s}$
12: **end procedure**

2) Blockchain Memory: We let L be the blockchain memory space, represented as the hastable $L : \{0,1\}^{256} \rightarrow \{0,1\}^N$, where $N >> 256$ and can store sufficiently-large documents. We assume this memory to be tamper-proof under the same adversarial model used in Bitcoin and other blockchains. To intuitively explain why such a trusted data-store can be implemented on any blockchain (including Bitcoin), consider the following simplified, albeit inefficient, implementation: A blockchain is a sequence of timestamped transactions, where each transaction includes a variable number of output addresses (each address is a 160-bit number). L could then be implemented as follows – the first two outputs in a transaction encode the 256-bit memory address pointer, as well as some auxiliary meta-data. The rest of the outputs construct the serialized document. When looking up $L[k]$, only the most recent transaction is returned, which allows update and delete operations in addition to inserts.

3) Policy: A set of permissions a user u grants service s, denoted by $POLICY_{u,s}$. For example, if u installs a mobile application requiring access to the user's location and contacts, then $POLICY_{u,s} = \{location, contacts\}$. It is instructive to note that any type of data could be stored safely this way, assuming the service will not subvert the protocol and label the data incorrectly. Safeguards to partially prevent this could be introduced to the mobile SDK, but in any case, the user could easily detect a service that cheats, as all changes are visible to her.

4) Auxiliary Functions: $Parse(x)$ de-seralizes the message sent to a transaction, which contains the arguments; $CheckPolicy(pk_{sig}^k, x_p)$, illustrated in Protocol 2, verifies that the originator has the appropriate permissions.

Protocol 2 Permissions check against the blockchain

1: **procedure** CHECKPOLICY(pk_{sig}^k, x_p)
2: $s \leftarrow 0$
3: $a_{policy} = \mathcal{H}(pk_{sig}^k)$
4: **if** $L[a_{policy}] \neq \emptyset$ **then**
5: $pk_{sig}^{u,s}, pk_{sig}^{s,u}, POLICY_{u,s} \leftarrow Parse(L[a_{policy}])$
6: **if** $pk_{sig}^k = pk_{sig}^{u,s}$ **or**
7: $(pk_{sig}^k = pk_{sig}^{s,u}$ **and** $x_p \in POLICY_{u,s})$ **then**
8: $s \leftarrow 1$
9: **end if**
10: **end if**
11: **return** s
12: **end procedure**

B. Blockchain Protocols

Here we provide a detailed description of the core protocols executed on the blockchain. Protocol 3 is executed by nodes in the network when a T_{access} transaction is received, and similarly, Protocol 4 is executed for T_{data} transactions.

As mentioned earlier in the paper, T_{access} transactions allow users to change the set of permissions granted to a service, by sending a $POLICY_{u,s}$ set. Sending the empty set revokes all access-rights previously granted. Sending a T_{access} transaction with a new compound identity for the first time is interpreted as a user signing up to a service.

Similarly, T_{data} transactions govern read/write operations. With the help of $CheckPolicy$, only the user (always) or the service (if allowed) can access the data. Note that in lines 9 and 16 of Protocol 4 we used shorthand notation for accessing the DHT like a normal hashtable. In practice, these instructions result in an off-blockchain network message (either read or write) that is sent to the DHT.

C. Privacy and Security Analysis

We rely on the blockchain being tamper-free, an assumption that requires a sufficiently large network of untrusted peers. In addition, we assume that the user manages her keys in a secure manner, for example using a secure-centralized wallet service. We now show how our system protects against adversaries compromising nodes in the system. Currently, we

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

Protocol 4 Storing or Loading Data

procedure HANDLEDATATX(pk^k_{sig}, m)
  $c, x_p, rw = Parse(m)$
  if $CheckPolicy(pk^k_{sig}, x_p) =$ `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

are less concerned about malicious services that change the protocol or record previously read data, as they are likely to be reputable, but we provide a possible solution for such behavior in section V-A.

Given this model, only the user has control over her data. The decentralized nature of the blockchain combined with digitally-signed transactions ensure that an adversary cannot pose as the user, or corrupt the network, as that would imply the adversary forged a digital-signature, or gained control over the majority of the network's resources. Similarly, an adversary cannot learn anything from the public ledger, as only hashed pointers are stored in it.

An adversary controlling one or more DHT nodes cannot learn anything about the raw data, as it is encrypted with keys that none of the nodes posses. Note that while data integrity is not ensured in each node, since a single node can tamper with its local copy or act in a byzantine way, we can still in practice minimize the risk with sufficient distribution and replication of the data.

Finally, generating a new compound identity for each user-service pair guarantees that only a small fraction of the data is compromised in the event of an adversary obtaining both the signing and encryption keys. If the adversary obtains only one of the keys, then the data is still safe. Note that in practice we could further split the identities to limit the exposure of a single compromised compound identity. For example, we can generate new keys for every hundred records stored.

V. Discussion of Future Extensions

In this section, we slightly digress to present possible future extensions to blockchains. These could play a significant role in shaping more mature distributed trusted computing platforms, compared to current state-of-the-art systems. More specifically, they would greatly increase the usefulness of the platform presented earlier.

A. From Storage to Processing

One of the major contributions of this paper is demonstrating how to overcome the public nature of the blockchain. So far, our analysis focused on storing pointers to encrypted data. While this approach is suitable for storage and random queries, it is not very efficient for processing data. More importantly, once a service queries a piece of raw data, it could store it for future analysis.

A better approach might be to never let a service observe the raw data, but instead, to allow it to run computations directly on the network and obtain the final results. If we split data into shares (e.g., using Shamir's Secret Sharing [23]), rather than encrypting them, we could then use secure Multi-party Computation (*MPC*) to securely evaluate any function [3].

In Figure 2, we illustrate how MPC might work with blockchains and specifically in our framework. Consider a simple example in which a city holds an election and wishes to allow online secret voting. It develops a mobile application for voting which makes use of our system, now augmented with the proposed MPC capabilities. After the online elections take place, the city subsequently submits their back-end code to aggregate the results. The network selects a subset of nodes at random and an interpreter transforms the code into a secure MPC protocol. Finally, the results are stored on the public ledger, where they are safe against tampering. As a result, no one learns what the individual votes were, but everyone can see the results of the elections.

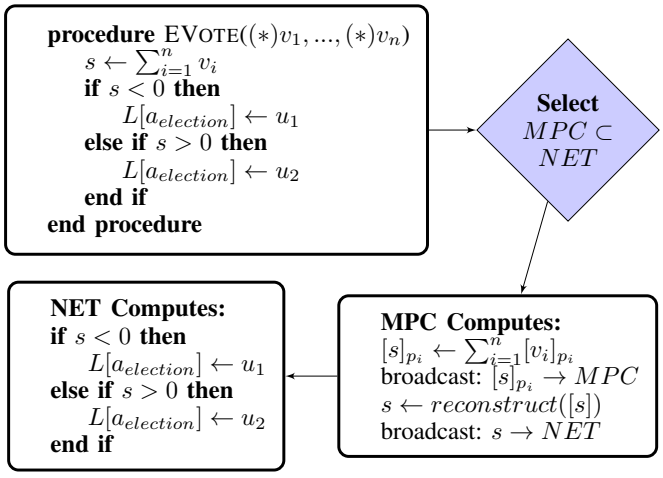

Fig. 2. Example of a flow of secure computation in a blockchain network. The top left block (EVote procedure) is the unsecure code, where the arguments marked in (*) are private and stored as shares on the DHT. The network selects a subset of nodes at random to compute a secure version of EVote and broadcasts the results back to the entire network, that stores it on the ledger.

B. Trust and Decision-Making in Blockchains

Bitcoin, or blockchains in general, assumes all nodes are equally untrusted and that their proportion in the collective decision-making process is solely based on their computational resources (known as the Proof-of-work algorithm) [17]. In other words – for every node n, $trust_n \propto resources(n)$ (probabilistically) decides the node's weight in votes. This leads to adverse effects, most notably vulnerability to sybil attacks, excessive energy consumption and high-latency.

Intuitively, Proof-of-Work reasons that nodes which pour significant resources into the system are less likely to cheat. Using similar reasoning we could define a new dynamic measure of trust that is based on node behavior, such that good actors that follow the protocol are rewarded. Specifically, we could set the trust of each node as the expected value of it behaving well in the future. Equivalently, since we are dealing with a binary random variable, the expected value is simply the probability p. A simple way to approximate this probability is by counting the number of good and bad actions a node takes, then using the sigmoid function to squash it into a probability. In practice, every block i we should re-evaluate the trust score of every node as –

$$trust_n^{(i)} = \frac{1}{1 + e^{-\alpha(\#good - \#bad)}}, \tag{3}$$

where α is simply the step size.

With this measure, the network could give more weight to *trusted* nodes and compute blocks more efficiently. Since it takes time to earn trust in the system, it should be resistant to sybil attacks. This mechanism could potentially attract other types of attacks, such as nodes increasing their reputation just to act maliciously at a later time. This might be mitigated by randomly selecting several nodes, weighted by their trust, to vote on each block, then taking the *equally-weighted* majority vote. This should prevent single actors from having too much influence, regardless of their trust-level.

VI. Conclusion

Personal data, and sensitive data in general, should not be trusted in the hands of third-parties, where they are susceptible to attacks and misuse. Instead, users should own and control their data without compromising security or limiting companies' and authorities' ability to provide personalized services. Our platform enables this by combining a blockchain, re-purposed as an access-control moderator, with an off-blockchain storage solution. Users are not required to trust any third-party and are always aware of the data that is being collected about them and how it is used. In addition, the blockchain recognizes the users as the owners of their personal data. Companies, in turn, can focus on utilizing data without being overly concerned about properly securing and compartmentalizing them.

Furthermore, with a decentralized platform, making legal and regulatory decisions about collecting, storing and sharing sensitive data should be simpler. Moreover, laws and regulations could be programmed into the blockchain itself, so that they are enforced automatically. In other situations, the ledger can act as legal evidence for accessing (or storing) data, since it is (computationally) tamper-proof.

Finally, we discussed several possible future extensions for blockchains that could harness them into a well-rounded solution for trusted computing problems in society.

References

[1] Scaling the facebook data warehouse to 300 pb, 2014.

[2] James Ball. Nsa's prism surveillance program: how it works and what it can do. *The Guardian*, 2013.

[3] Michael Ben-Or, Shafi Goldwasser, and Avi Wigderson. Completeness theorems for non-cryptographic fault-tolerant distributed computation. In *Proceedings of the twentieth annual ACM symposium on Theory of computing*, pages 1–10. ACM, 1988.

[4] EUROPEAN COMMISSION. *Commission proposes a comprehensive reform of data protection rules to increase users' control of their data and to cut costs for businesses*. 2012.

[5] Yves-Alexandre de Montjoye, César A Hidalgo, Michel Verleysen, and Vincent D Blondel. Unique in the crowd: The privacy bounds of human mobility. *Scientific reports*, 3, 2013.

[6] Yves-Alexandre de Montjoye, Erez Shmueli, Samuel S Wang, and Alex Sandy Pentland. openpds: Protecting the privacy of metadata through safeanswers. *PloS one*, 9(7):e98790, 2014.

[7] Cynthia Dwork. Differential privacy. In *Automata, languages and programming*, pages 1–12. Springer, 2006.

[8] Jon Evans. Bitcoin 2.0: Sidechains and ethereum and zerocash, oh my!, 2014.

[9] Craig Gentry. Fully homomorphic encryption using ideal lattices. In *STOC*, volume 9, pages 169–178, 2009.

[10] Vindu Goel. Facebook tinkers with users' emotions in news feed experiment, stirring outcry. *The New York Times*, 2014.

[11] Federal Information and Processing Standards. FIPS PUB 180-4 Secure Hash Standard (SHS). (March), 2012.

[12] Don Johnson, Alfred Menezes, and Scott Vanstone. The elliptic curve digital signature algorithm (ecdsa). *International Journal of Information Security*, 1(1):36–63, 2001.

[13] Michael Lesk. How much information is there in the world?

[14] Ninghui Li, Tiancheng Li, and Suresh Venkatasubramanian. t-closeness: Privacy beyond k-anonymity and l-diversity. In *ICDE*, volume 7, pages 106–115, 2007.

[15] Ashwin Machanavajjhala, Daniel Kifer, Johannes Gehrke, and Muthuramakrishnan Venkitasubramaniam. l-diversity: Privacy beyond k-anonymity. *ACM Transactions on Knowledge Discovery from Data (TKDD)*, 1(1):3, 2007.

[16] Petar Maymounkov and David Mazieres. Kademlia: A peer-to-peer information system based on the xor metric. In *Peer-to-Peer Systems*, pages 53–65. Springer, 2002.

[17] Satoshi Nakamoto. Bitcoin: A peer-to-peer electronic cash system. *Consulted*, 1(2012):28, 2008.

[18] Arvind Narayanan and Vitaly Shmatikov. How to break anonymity of the netflix prize dataset. *arXiv preprint cs/0610105*, 2006.

[19] Juan Perez. Facebook, google launch data portability programs to all, 2008.

[20] Rt.com. Obama announces legislation protecting personal data, student digital privacy, 2015.

[21] K Schwab, A Marcus, JO Oyola, W Hoffman, and M Luzi. Personal data: The emergence of a new asset class. In *An Initiative of the World Economic Forum*, 2011.

[22] ScienceDaily. Big data, for better or worse: 90% of world's data generated over last two years. 2013.

[23] Adi Shamir. How to share a secret. *Communications of the ACM*, 22(11):612–613, 1979.

[24] Latanya Sweeney. k-anonymity: A model for protecting privacy. *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems*, 10(05):557–570, 2002.