Secure Credit Reporting on the Blockchain

Amir Kafshdar Goharshady
IST Austria
Klosterneuburg, Austria
amir.goharshady@ist.ac.at

Ali Behrouz
Department of Computer Engineering
Sharif University of Technology
Tehran, Iran
abehrouz@ce.sharif.edu

Krishnendu Chatterjee
IST Austria
Klosterneuburg, Austria
krishnendu.chatterjee@ist.ac.at

***Abstract*—We present a secure approach for maintaining and reporting credit history records on the Blockchain. Our approach removes third-parties such as credit reporting agencies from the lending process and replaces them with smart contracts. This allows customers to interact directly with the lenders or banks while ensuring the integrity, unmalleability and privacy of their credit data. Additionally, each customer has full control over complete or selective disclosure of her credit records, eliminating the risk of privacy violations or data breaches. Moreover, our approach provides strong guarantees for the lenders as well. A lender can check both correctness and completeness of the credit data disclosed to her. This is the first approach that can perform all credit reporting tasks without a central authority or changing the financial mechanisms*[*]*.**

***Index Terms*—Credit Reporting, Smart Contracts, Blockchain**

1. Introduction and Preliminaries

In this section, we first provide a high-level overview of both smart contracts and credit reporting services. Then, we discuss some of the problems that currently exist in real-world credit reporting and argue that these can be mitigated by decentralization and migrating to smart contracts.

Blockchain. Blockchain was initially used as a means to achieve global consensus about peer-to-peer cryptocurrency transactions in Bitcoin [2]. However, the technology itself is capable of much more than just verifying transactions. Specifically, one can include scripts in transactions, forcing a consensus about the outputs of these scripts. Bitcoin allows simple scripting in a Forth-like loop-free language [3]. A script in a Bitcoin transaction is essentially a program that sets the conditions one must satisfy in order to use the currency units stored in that transaction. For example, a script might ask for a digital signature.

Ethereum and Smart Contracts. Ethereum is a cryptocurrency that allows stateful scripts of arbitrary, i.e. Turing-complete, complexity [4]. It provides an ecosystem for the development of decentralized applications, called smart contracts, that are executed and verified by the whole Ethereum network. A smart contract can be created by anyone and is stored in a bytecode format on the Blockchain. After its creation, the contract can save data in its own dedicated storage and hold, receive and transfer funds (cryptocurrency units) from/to other people or contracts. It can also interact with other contracts and even create new ones. However, the state and actions of the contract are all controlled by its code and subject to consensus using the Blockchain protocol. After its deployment, one can only interact with a contract by calling its functions which perform actions as programmed by its creator. These characteristics, and the inherent lack of a centralized authority in the Blockchain, make smart contracts ideal for implementing a variety of unbreakable financial agreements. For example, a smart contract called BitHalo replaces trusted third-parties and provides escrow services [5]. See [1] for more examples.

Credit Reporting. A credit report is a document that includes data regarding a person's history of managing credit. This data is used to assess the creditworthiness of the individual. It usually contains the following information [6]:

- Identifying information, such as the name, address and social security number, of the individual.
- Information reported to the credit reporting agency by creditors, such as banks, regarding details of current and past loans, leases, credit report requests, bills, etc. We refer to each of these as a *credit account*.
- Public records such as bankruptcy information.

Credit Reporting Industry. CRAs are the companies that gather credit report information. They compile credit data and help future lenders decide about extending credit. There are three major CRAs in the US. In 2003, they issued 2 million credit reports each day and each of them was estimated to hold data about roughly 1.5 billion credit accounts belonging to 190 million individuals [6]. One of them, Equifax, reported a revenue of nearly $850 million in the third quarter of 2017 [7].

Problems with Credit Reporting. The fact that CRAs are collecting and storing vast amounts of sensitive data about hundreds of millions of people is concerning. To address these concerns, laws are passed to ensure the rights of individuals to privacy and fair treatment. One example is the US Fair Credit Reporting Act (FCRA) of 1970. However, there are a variety of problems that cannot be addressed by

[*]An extended version of this article is accessible at [1].

regulation alone. These include:

- *Long Update Intervals.* CRAs generally receive information from creditors once a month and it takes them up to seven days to update the records [6].
- *Identification Problems.* The data received by the CRAs does not always include uniquely-identifying information and might be erroneously attributed to the wrong individual [8]. Another related aspect of this problem is identity theft. It was estimated that 12 percent of Americans were victims of identity theft in the 5-year period ending in 2003 [9].
- *Errors and Inconsistency.* Reports stored by different CRAs can be inconsistent or contradictory [8]. Moreover, it is estimated that as many as a third of all credit reports might contain errors that can lead to denial of access to credit [6], [10].
- *Endemism.* Credit data is usually tied to a single country or jurisdiction. The CRAs cannot access foreign credit information [6]. When an individual relocates, her credit data is effectively erased.
- *Data Breaches.* A major source of discomfort is the possibility of data breaches and unauthorized access to the sensitive credit report information. In a famous catastrophic case in 2017, hackers stole sensitive data of 143 million people from Equifax [11].

Our Contribution. In this paper, we propose an approach based on smart contracts that can remove the CRAs from the lending process and fixes all the problems mentioned above. In our approach (i) data updates take only a few seconds, (ii) identification problems are entirely avoided, (iii) there is no possibility of inconsistency, (iv) credit reports can be used globally and (v) all sensitive information is secured by cryptography. We also give individuals full control over their credit report, allowing them to disclose all or any part of it to others. From the creditors' point-of-view, we guarantee that the report is correct and not editable by its owner and that the creditor can easily check to ascertain that it includes all the data in a requested time-frame.

We now provide a high-level overview of our approach. We first recall the main concepts of encryption, decryption and digital signatures and then proceed with an intuitive description of our method. A more formal treatment is provided in the next sections.

Asymmetrical Cryptography. We assume basic familiarity with asymmetrical and public-key cryptography, as introduced e.g. in [12]. Formally, we use pairs of keys of the form (K, k) for encryption, decryption and digital signatures. The public key is denoted as K and its corresponding private key as k. One can encrypt data using K and then the encrypted data can only be decrypted if one knows k. Similarly, one can sign a piece of data using k and this signature is verifiable by anyone who has access to the data and K. In particular, a function call in a smart contract always includes the public key K and is signed by the private key k. This means that anyone can see the function call data and its caller by reading the Blockchain but no one can make a fake function call on behalf of another person unless they have access to her private key.

Underlying Principles of Our Approach. We achieve the above by employing the following techniques:

(i) *Identity Management.* We use a decentralized identity management and certification system in which a borrower's identity can be certified by lenders.

(ii) *Data Encryption.* We store the credit data in an encrypted format, using asymmetrical encryption. Only the owner and creator of a record can decrypt it.

(iii) *Links Encryption.* We chain the records belonging to each individual in a linked list whose pointers are also encrypted. Hence, not only one cannot read a record without authorization, but it is also impossible to find the owner of a given record.

(iv) *Fraud Prevention.* We use digital signatures and asymmetrical cryptography to avoid fraud. The simplified intuition is that a credit record can be first signed by the creditor and then encrypted using a key pair that is shared with the customer. Then, when another creditor wants to see the record, the customer can decode it and the creditor can check the previous creditor's signature to make sure the customer has not altered the record.

The main novelty of our approach is a combination of these ideas that achieves secure credit reporting on the Blockchain. To the best of our knowledge, this is the first method that can reliably perform all credit reporting tasks without trusted third-parties or changing the financial mechanism of credit reporting.

Organization. Our approach consists of three distinct protocols, each realized by a different smart contract. In Section 2, we present our solution for identity management. Section 3 explains how we handle credit accounts. This is followed by our public records protocol in Section 4. Section 5 provides a short report on a proof-of-concept implementation that is publicly available. We discuss some limitations of our approach in Section 6 and finally, Section 7 concludes.

2. Identity Management Protocol

One of the main issues in credit reporting, as in many other distributed applications, is identity management. There are two important aspects to this issue: first, one should not be able to masquerade as another person, i.e. commit identity theft, and second, one should not be able to use more than one identity. Note that in a cryptocurrency setting individuals having multiple identities do not pose a problem, given that this does not entail any benefit. However, disjoint credit reports for a single person should not be allowed.

A simple solution is to create one or several central authorities that check real-world identities and issue certificates of their validity. This is the solution used, for example, for checking valid HTTPS signatures [13]. It is also commonly used for managing the identities of banks, institutions and public authorities. In this paper, we assume that such entities' identities can be verified in this manner. However, the same approach is not desirable for individual credit customers, because it puts much power in the hands of the certificate issuers and they can, at least in theory, bar one from credit by refusing to certify.

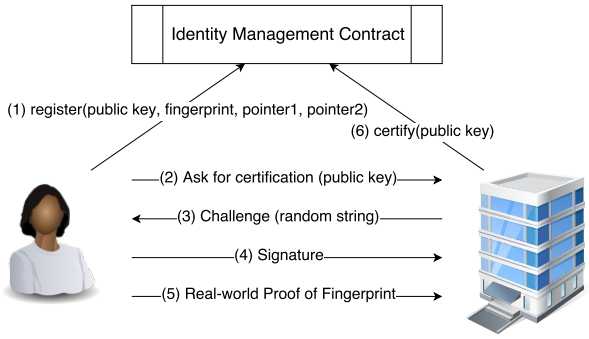

Figure 1. Interactions between an individual, an institution and the identity management contract. Numbers denote the order of actions.

Our proposal is to let the lenders act as certificate authorities. Concretely, we allow anyone to issue a certificate, but we expect the lenders, who are typically banks and financial institutions, to only take into account certificates issued by other banks or institutions that they already trust. Given that the lenders trust data sent by other lenders to the CRAs, which includes identifying information about the owners of credit accounts, it is expected that they agree to accept this same information directly, i.e. without the CRAs as middlemen, too. While this approach might lead to a situation where a few banks perform most of the certifications, this is not considered to be a problem, since no group of institutions have a monopoly on certification and every lender who is willing to extend credit to an individual can also certify her identity.

Data Fields of the Identity Management Contract. We now formalize our identity management protocol. It is realized by a single instance of a smart contract that keeps track of every individual by storing the following data:

- The *public key* used by the individual.
- *Fingerprint.* A unique identifier that can be used in real world to check the individual's identity. This can be biometric data or any other data that is unique to the individual. Our approach is not dependent on the exact standard that is used for creating fingerprints, but they should be standardized. If this data is sensitive, one can store a hashed version of it. For example, we can use a hashed version of the individual's country of nationality, appended with her national id number.
- *Two Pointers.* A pointer to the first public record of the individual and another one to her first credit account. These will be formalized in the next sections.
- *Certificates.* A list of public keys who have verified this identity in the real world.

Functions of the Identity Management Contract. We now describe how our identity management smart contract works. This is summarized in Figure 1. Anyone can register in this contract by calling the `register` function and providing her own desired public key and (possibly fake) fingerprint. The contract even allows several public keys to be registered as corresponding to the same fingerprint. After a public key and its corresponding fingerprint are added to the contract, anyone can call the function `certify` and announce that they have checked an identity in the real world and would like to certify it. In this case, the caller's public key is added to the list of certificates. There is also a `decertify` function that can be used to revoke the certification.

Safety against Sybil Attacks. One can create as many fake identities and certify them with as many self-created keys as she wishes. The lenders would only consider certificates from other trusted lenders or institutions. Such an institution would (i) ask the individual to sign a random piece of data using the private key corresponding to the desired public key to ensure that she has access to it, (ii) require real-world verification of the fingerprint, and (iii) require that no other public key is already certified as corresponding to the same fingerprint by another trusted institution.

Legal Guarantees. Note that the institutions, such as banks, have publicly announced public keys and will be subject to legal action should they provide false certifications. The process is also uniquely transparent, given that all changes to the contract are permanently recorded in the Blockchain. An individual can ask each lender she deals with to certify her identity so that the respective credit account is also trusted by future lenders.

Privacy. Our protocol preserves user privacy. The fingerprint is associated with a public key that does not appear in the credit accounts, ensuring that even having access to a person's fingerprint cannot be used to extract information about their non-public credit records. In the next sections, we will show that an attacker with access to the Blockchain cannot read data about the records, such as account details, and is even unable to infer the owner of a given record.

3. Credit Accounts Protocol

We now turn to the core of our approach, which is a protocol for storing credit accounts' data. We introduce a smart contract for modeling credit accounts. Each account is realized by one instance of this contract. This is in contrast to Section 2 where all identities were stored in a single instance of the identity management contract.

As mentioned earlier, we rely on asymmetric (public-key) cryptography. To achieve the desired level of security, we will introduce several new keys in this section. Therefore, to avoid confusion, we use the term "true identity" to refer to the key pair which is publicly known to belong to an institution. Similarly, an individual's true identity is the key pair with which she registers in the identity management protocol and for which she obtains certificates. We use K for public keys and k for private keys.

We store a singly linked list of each individual's credit accounts, with each account providing a pointer to the next (Figure 2). Note that in Ethereum each deployed instance of a smart contract is uniquely addressable and therefore these pointers are well-defined. The identity management contract provides a pointer to the first credit account. Moreover, these pointers are encrypted, as explained below, and hence they can only be traversed if the individual owner allows it.

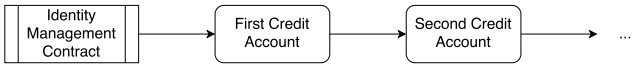

Figure 2. Each credit account is stored in its own instance of the credit account contract. The arrows denote encrypted pointers.

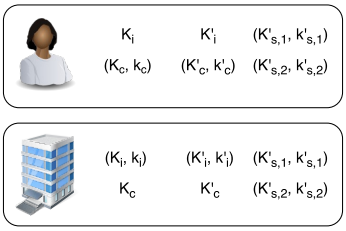

Figure 3. Key distribution prior to deployment of a Credit Account Contract

We now define the data in a credit account contract and the process for its creation, management and use.

Key Generation. Let the institution's true identity be (K_i, k_i) and the customer's true identity (K_c, k_c). When, after verifying a customer's identity and credit record, an institution agrees to extend credit to a customer, they ask her to create a new key pair (K'_c, k'_c), called customer's account-specific keys. The institution in turn creates its own account-specific keys (K'_i, k'_i). Then, each side provides the other side with their account-specific public key. Finally, they create and fully exchange two other pairs of keys $(K'_{s,1}, k'_{s,1}), (K'_{s,2}, k'_{s,2})$, which we call account-specific shared keys. The keys are distributed as in Figure 3.

Contract Creation. At this point, the institution creates a new instance of the credit account contract and publishes it on the Blockchain. Figure 4 shows the data stored in this contract and the conditions enforced by the contract for changing this data. The contract stores public keys of the customer and the institution, i.e. K'_c and K'_i. These are set at the beginning and are not changeable afterwards. The contract does not store true identities, but uses contract-specific public keys instead. All function calls are also performed using contract-specific keys. The reason behind this is that anyone has access to the data stored on the Blockchain and one must not be able to read the true identities using publicly available data. The contract also has an expiry time which can be changed only if both parties agree.

Commitment. After deployment, both parties must commit to the contract by verifiably connecting it to their true identity. The institution does this by signing the contract address, K_i and K_c using its true identity and adding the signature to the contract. At this point, the customer can check the signature. If the check passes, she adds the contract to her record by letting her last account's `Next Account` field point to this contract by storing its address encrypted using $K'_{s,2}$. Note that the `Next Account` field can be changed only once and hence the contract cannot be removed from the customer's report when added. The institution can now check that the contract is in customer's report using $k'_{s,2}$.

Credit Report Data. Finally, the institution can change the contents of the field `Data` until the expiry time. It can store all the relevant data about this account that should appear in a credit report. This data is always encrypted using $K'_{s,1}$ and is hence accessible to both the institution and the customer, who know $k'_{s,1}$, but not to anyone else. If the data happens to be too big, one can store it in an external service, such as IPFS [14] and then fill the `Data` field with an address/identifier and hash of the original data.

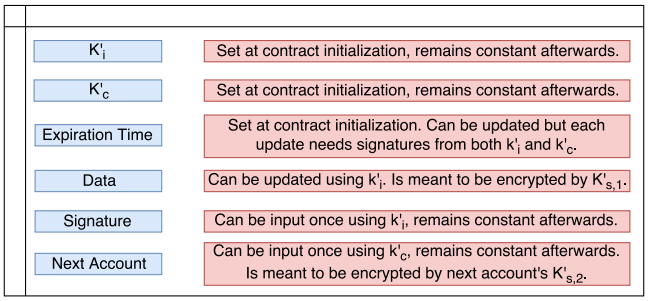

Figure 4. Data fields and constraints in a Credit Account Contract

Reading a Credit Report. When another institution wants to read customer data, it would need the values of $k'_{s,2}$ for each of the contracts to be able to decrypt the links and traverse the linked list. These can only be provided by the customer. Hence, one cannot find out which accounts belong to an individual, unless that individual allows access. When access is granted, the institution can easily find out when it reaches the end of a report given that the `Next Account` field is only empty at the end of the linked list. The institution can also see the beginning time of a contract by looking up the number of the Blockchain block where the contract was first created. Expiration times of the credit accounts are publicly visible on the Blockchain, but not their data. Should the customer decide to allow the institution to read a contract, she can provide them with the contract-specific $k'_{s,1}$ to access the `Data` field and with the lender's true identity, K_i, to verify the signature.

An individual can add as many credit accounts as she wishes to her linked list, acting as both the institution and the customer. This can be used to initialize the linked list by an account when creating an identity, and also to resist any attempt to find out the true number of accounts.

4. Public Records Protocol

Our protocol for storing public records is similar to the one we described for credit accounts. However, in this case the protocol becomes much simpler, because unlike credit accounts, public records can be made without the consent of their individual owners. Similar to the previous section, we store public records in a singly linked list. Each record is an instance of the public record contract. As previously mentioned in Section 2, there is a pointer from an individual's identity to her first public record, which can be created by herself.

Unlike credit accounts, the pointers used to connect public records are not encrypted. This allows anyone to

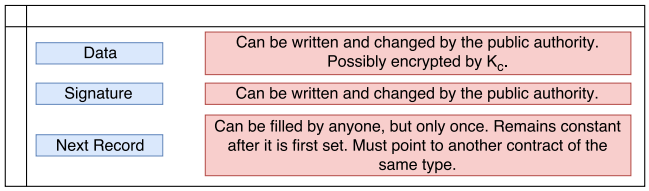

Figure 5. Data fields and constraints in a Public Record Contract

follow the list of public records corresponding to an identity. Anyone can add a new public record to the end of any of these lists. This is not problematic, given that lenders will only take the records issued by real public institutions into account. Simply, each record is either added using an unknown identity, in which case it is spam and ignored[†], or by an official identity, in which case it is either correct or can be corrected by the same authority. Again, note that all changes to the contracts are permanently saved on the Blockchain and that official authorities are bound by legal responsibilities and cannot simply issue false records.

We now formalize this. Figure 5 shows the data fields of a public record contract and their constraints.

Contract Creation. The public authority creates an instance of this contract and publishes it on the Blockchain. The authority has access to the individual's fingerprint and can hence add the record to linked lists corresponding to all identities that have that fingerprint. To do so, the authority follows the `Next Record` pointers until it reaches the end of the linked list, and then sets the final `Next Record` to point to the new instance of the contract. Anyone can set the value for `Next Record`, but (i) it can be filled only once and (ii) it must keep the linked list valid and extensible. We refer to the latter condition as "validity".

Credit Report Data. The other two data fields in this contract, `Data` and `Signature`, are under complete control of its issuer. `Data` is meant to contain any relevant information that should be considered part of the credit report. The authority can decide whether to fill this data without encryption, hence allowing public access to it, or encrypt it using K_c, so that it is only accessible by the individual owner herself. In the latter case, the authority signs the original unencrypted `Data` and stores this signature in the contract. This ensures that the individual owner can both read and prove what is saved in `Data` and is the only person, other than the public authority, who can perform these actions.

Reading a Credit Report. When an institution decides to read the public records of an individual, it simply follows the linked list, ignoring spam. In case it faces an encrypted entry by a trusted public authority, it asks the individual owner to decrypt the `Data` field and provide the decrypted text. It then checks the signature to make sure that the text was not changed by the owner.

[†]Spamming is not free given that one has to pay for its gas fees. This is Ethereum's solution to combat spam and it naturally extends to our contracts. On the other hand, when reading the records, one can differentiate spam entries pretty fast, by simply checking the identity of their signatures. Note that reading the blockchain is free but writing to it is not.

Importance of Validity. When dealing with credit accounts, the pointers used for our linked list were filled by the individual who owned them and there was no fear that she might intend to destruct the whole linked list. Also, the signatures provided by the institutions guaranteed that one cannot add another person's record to her linked list without getting caught. However, in the case of public records, anyone can add a new element to the linked list and fill the `Next Record` fields. These fields remain immutable after they are first filled. So, a natural attack would be to fill them with invalid pointers, i.e. pointers that do not hold the address of a valid contract of the same type. This will make it impossible for others to keep adding records. Another malicious behavior is adding the same instance of a record to the linked lists belonging to two different individuals. This will merge the two lists.

Enforcing Validity. To avoid the attacks described above, we do not allow the individuals to create instances of our public record contract directly. Instead, we develop a so-called "factory" contract that can be called by anyone to create valid instances of the public record contract. The factory contract also keeps track of the addresses of all valid public record contracts instantiated using it and whether they have been added to a linked list. On the other hand, each such instantiated contract includes an immutable pointer to the parent factory contract. When a new contract is being added to the linked list, it is first checked against the factory contract to ensure it respects validity. See [1] for details.

Deanonymization. The fact that public records are not encrypted means that they can be used to deanonymize users. For example, public records of bankruptcy often include names of individuals and their national identity numbers, which might be the same as fingerprints. However, the only additional data that can be inferred by such deanonymization is the individual's public key K_c. As mentioned before, this key is not saved in any of the credit account contracts and cannot be used to infer any non-public information about the individual. Note that the public records themselves are, and should be, accessible to everyone.

5. Implementation

We have implemented our approach in Solidity to demonstrate the feasibility of the ideas and structures that we suggest. A proof-of-concept implementation, together with instructions for its deployment and testing, is available at `pub.ist.ac.at/~akafshda/credit-reporting`.

Our implementation is entirely loop-free and all of its function calls terminate after executing a small (constantly-bounded) number of instructions. Hence, our gas cost, i.e. the cost one must pay for execution of commands in Ethereum smart contracts [4], is very little.

6. Limitations

We discuss some limitations of our approach and ideas to address them. See [1] for a comparison with related work.

Inherited Limitations. The goal of our approach is to remove the CRAs from the credit reporting process, allowing the same financial mechanisms that are currently established to run without a middleman. This means that our approach essentially inherits any limitation of the traditional centralized credit reporting that is not due to the CRAs. Particularly, if an individual has two provable identities in the real world, e.g. two distinct names and national identity numbers, then she can sign up in our identity management contract twice and obtain certificates for both. This attack is not dependent on the lack of CRAs and is also possible under the current credit reporting systems.

Cryptographic Primitives. The security of our approach is dependent on the security of the cryptographic primitives that are used. Any data saved on the blockchain is permanent. In several of the above protocols, data encryption is used in order to restrict public access. If/when the underlying ciphers are broken, this data can be recovered. Therefore, it is advisable to refrain from saving the actual credit data in smart contracts, but instead rely on saving its hash. This way the data would be provable, but cannot be obtained even if the cipher breaks. The downside to this is that the individual will have to keep safe copies of the original data and can only use our approach for proving her record.

7. Conclusion

In this paper, we presented the first solution for secure credit reporting with no third-parties. In Section 1 we identified five problems with current systems of credit reporting that can be avoided by migrating to the blockchain. We review how our approach solves these problems:

- *Long Update Intervals.* Each update is done via a single function call in one of the smart contracts. Hence, it takes a few seconds to be added to the Ethereum blockchain, and after a few minutes one can be sure that it will not be reverted.
- *Identification Problems.* The certification protocol of Section 2 ensures only valid real-world identities will be trusted by the institutions and that each real-world identity is represented by a single public key K_c.
- *Errors and Inconsistency.* Inconsistency can only be caused by forks in the blockchain and disappears as soon as the fork is resolved. Wrong data added by an institution can always be fixed by the same institution[‡]. The source of such data can be provably ascertained. Hence, the institution is legally bound to fix it.
- *Endemism.* Using the Ethereum blockchain, the contracts and their data can be used globally.
- *Data Breaches.* There is no central authority possessing all the data. Each credit account is secured by its own keys. Hence, a large-scale breach is impossible unless the underlying cryptographic ciphers break.

[‡]Note that while the contents of the blockchain are immutable, the values of contract variables are not. The blockchain saves the sequence of changes to these values. Hence, once an error is fixed, its history remains in the blockchain.

Acknowledgments. We are thankful to the reviewers for raising points that significantly improved this article. The research was partially supported by Vienna Science and Technology Fund (WWTF) Project ICT15-003, Austrian Science Fund (FWF) NFN Grant No S11407-N23 (RiSE/SHiNE) and ERC Starting grant (279307: Graph Games). The first author is supported by an IBM PhD Fellowship.

References

[1] A. K. Goharshady, A. Behrouz, and K. Chatterjee, "Secure credit reporting on the blockchain," *arXiv preprint arXiv:1805.09104*, 2018.

[2] S. Nakamoto, "Bitcoin: A peer-to-peer electronic cash system," 2008.

[3] Bitcoin Wiki, "Script." [Online]. Available: https://en.bitcoin.it/wiki/Script

[4] G. Wood, "Ethereum: A secure decentralised generalised transaction ledger," *Ethereum Project Yellow Paper*, 2014.

[5] D. Zimbeck, S. Donato, A. Hahn, P. Sloin, and G. Meacci, "Bithalo, mother of smart contracts and a decentralized market for everything." [Online]. Available: http://bithalo.org

[6] R. B. Avery, P. S. Calem, G. B. Canner, and R. W. Bostic, "An overview of consumer data and credit reporting," *Federal Reserve Bulletin*, vol. 89, p. 47, 2003.

[7] Equifax, "Equifax releases third quarter results," Nov. 2017. [Online]. Available: https://investor.equifax.com/news-and-events/news/2017/11-09-2017-211550295

[8] Consumer Federation of America and the National Credit Reporting Association, "Credit score accuracy and implications for consumers," Tech. Rep., 2002.

[9] C. M. Kahn and W. Roberds, "Credit and identity theft," *Journal of Monetary Economics*, vol. 55, no. 2, pp. 251–264, 2008.

[10] J. Golinger and E. Mierzwinski, *Mistakes do happen: Credit report errors mean consumers lose*. Washington Public Interest Research Group, 1998.

[11] Federal Trade Commission, "The equifax data breach: What to do," 2017. [Online]. Available: https://www.consumer.ftc.gov/blog/2017/09/equifax-data-breach-what-do

[12] J. Hoffstein, J. C. Pipher, and J. H. Silverman, *An introduction to mathematical cryptography*. Springer, 2008.

[13] Z. Durumeric, J. Kasten, M. Bailey, and J. A. Halderman, "Analysis of the https certificate ecosystem," in *Proceedings of the 2013 conference on Internet measurement conference*. ACM, 2013, pp. 291–304.

[14] J. Benet, "IPFS - content addressed, versioned, p2p file system," *IPFS Whitepaper*, 2018. [Online]. Available: https://github.com/ipfs/papers/raw/master/ipfs-cap2pfs/ipfs-p2p-file-system.pdf