The Limit of Blockchains: Infeasibility of a Smart Obama-Trump Contract*

Yongge Wang
Department of SIS, UNC Charlotte, USA
and
Qutaibah m. Malluhi
Department of Computer Science and Engineering, Qatar University, Qatar.

May 13, 2018

Abstract

Blockchains have become a buzzword and many blockchain proponents believe that smart contract is a panacea to redefine the digital economy. The community has a misconception that any kind of contracts could be implemented as a blockchain smart contract. There is no doubt that Turing-complete scripting languages in blockchain techniques such as Ethereum can be used to draft many important smart contracts. However, digital economy is much more than Turing-complete smart contracts. Many protocols/contracts in our daily life could not be implemented using Turing-complete smart contracts. As an example, we formulate an Obama-Trump contract and show that this kind of contract could not be implemented using blockchain smart contract techniques. It is straightforward to observe that many contracts in our daily life could be described in terms of an Obama-Trump contract. As a background discussion, we also give a comprehensive review of historical cryptographic currency techniques, bitcoin smart contract techniques, and Turing-complete smart contract techniques in modern blockchains.

1 Cryptographic currency

When Internet becomes more and more invasive in our daily life, it will be convenient to have a digital payment system or to design a digital currency for our society. Generally it is easy to design an electronic cash system using Public Key Infrastructure (PKI) systems. However, PKI based electronic cash is easy to trace. Though bank notes could be traced using sequence numbers, there is no convenient infrastructure to trace bank note sequence numbers back to users. Thus bank notes maintain sufficient anonymity.

For an electronic cash system, it must avoid double spending and it is preferred to be non-traceable. Also it is preferred to be able to make small payments of a few cents on line. Such kind of electronic cash systems could be designed using Chaum's blind signatures for untraceable payments [3] that was invented in 1983. Assume that the bank has an RSA public key (e, N) and a private key d. In order for Alice to withdraw \$10 from her bank account and convert it to a digital coin m of \$10, they carry out the following protocol.

- Alice chooses a random number r and computes $m' = m \cdot r^e (\text{mod } N)$.
- The bank generates a signature $s' = (m')^d$ on m'.
- Alice calculates a signature s on m as $s = s' \cdot r^{-1} = (m \cdot r^e)^d \cdot r^{-1} = m^d$.
- Alice spends (m, s) as \$10 on line while bank cannot link this coin m to Alice's account.

There are various challenges to the above blind signature based electronic cash system. The first challenge is what happens if Alice asks the bank to sign $m' = 100 \cdot r^e (\text{mod } N)$ instead of $m' = 10 \cdot r^e (\text{mod } N)$? This challenge could be resolved by either requiring that all coins have the same value or by using the following probabilistic approach:

*The work reported in this paper is supported by Qatar Foundation Grant NPRP X-063-1-014

- Alice generates 100 blind coins: $m'_i = m_i \cdot r_i^e (\text{mod } N)$ for $i = 1, \cdots, 100$.
- The bank randomly selects $m'_{j_1}, \cdots, m'_{j_{99}}$.
- Alice reveals the values m_{j_i}, r_{j_i} to the bank for $i = 1, \cdots, 99$.
- The bank issues a signature on the remaining m' only if the $m_{j_i} = 10$ and $m'_{j_i} = m_{j_i} \cdot r_{j_i}^e (\text{mod } N)$ for $i = 1, \cdots, 99$.

The second challenge for Chaum's blind signature based electronic cash system is that a seller must contact the bank to make sure that the coin m has not been spent yet before accepting this coin m from Alice. This requires that the bank remains online all the time. Chaum, Fiat, and Naor [4] constructed an electronic cash that does not need the bank to be online. Let H_1, H_2 be hash functions and k be a fixed even integer. Assume that Alice has an account u with bank and bank keeps a counter number v for Alice. In order for Alice to get a digital coin from the bank, the following steps are carried out:

- Alice chooses random a_i, c_i, d_i and r_i for $1 \leq i \leq k$.
- Alice sends k blind candidates $B_i = r_i^e \cdot H_1(x_i, y_i)$ to the bank where
$$x_i = H_2(a_i, c_i)$$
$$y_i = H_2(a_i \oplus (u \| (v+i)), d_i)$$
- The bank chooses a random subset $R \subset \{1, \cdots k\}$ of size $k/2$ and sends R to Alice.
- Alice reveals a_i, c_i, d_i and r_i for all $i \in R$.
- Bank signs $S = \prod_{i \notin R} B_i^d$, deducts the dollar from Alice account, and increases v by k.
- Alice extracts coin $C = S \cdot \left(\prod_{i \notin R} r_i\right)^{-1} = \prod_{i \notin R} \left(H_1(x_i, y_i)\right)^d$.

When Alice wants to make a payment to Bob, Alice sends C to Bob. Assume that

$$\{i_1, \cdots, i_{k/2}\} = \{1, 2, \cdots, k\} \setminus R.$$

Bob sends random bits $z_{i_1}, \cdots, z_{i_{k/2}}$ to Alice. For $j = 1, \cdots, k/2$, Alice responds as follows:

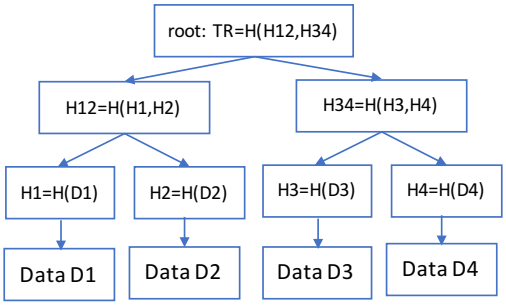
1. If $z_{i_j} = 0$, then Alice sends a_{i_j}, c_{i_j} and y_{i_j} to Bob. In this case, Bob is able to compute
$$H_1\left(x_{i_j}, y_{i_j}\right) = H_1\left(H_2(a_{i_j}, c_{i_j}), y_{i_j}\right).$$
2. If $z_{i_j} = 1$, then Alice sends $x_{i_j}, a_{i_j} \oplus (u \| (v + i_j))$, and d_{i_j} to Bob. In this case, Bob is able to compute
$$H_1\left(x_{i_j}, y_{i_j}\right) = H_1\left(x_{i_j}, H_2(a_{i_j} \oplus (u \| (v + i_j)), d_{i_j})\right).$$

After receiving these values, Bob is able to verify that C is a signature on the message $\prod_{j=1}^{k/2} H_1(x_{i_j}, y_{i_j})$.

In the above transaction process, Alice's bank does not need to be online. In order for Bob to cash the coin C at Alice's bank, Bob sends the coin C together with Alice's response to Alice's bank. One may wonder if Alice's bank is not online, how can we avoid double spending? If Alice spends the same coin both at Bob's shop and at Charlie's shop, then the challenge sequences $z_{i_1}, \cdots, z_{i_{k/2}}$ from Bob and Charlie are different with high probability. Assume that the challenge bit $z_{i_1} = 0$ for Bob and $z_{i_1} = 1$ for Charlie. Then Alice has revealed $a_{i_1}, c_{i_1}, y_{i_1}, x_{i_1}, a_{i_1} \oplus (u \| (v + i_1))$, and d_{i_1}. That is, Alice's account number u could be recovered from these revealed values. In other words, if Alice double spends, her identity will be revealed.

Many other non-PKI based digital cash systems have been proposed in the literature also. For example, Rivest and Shamir [15] proposed the PayWord and MicroMint payment schemes. In the PawWord scheme, Alice computes a sequence of binary strings $w_0, w_1, w_2, \cdots, w_n$ such that $w_i = H(w_{i+1})$ where H is a secure cryptographic hash function. Alice then commits w_0 to bank which cannot be spent. Assume that each payment is one cent, then the i-th cent is spent as (i, w_i). In the MicroMint scheme, there is a central broker to mint the coins. For example, in order for the broker to mint 2^{30} coins, it will use an array of 2^{30}. The broker will repeatedly hash randomly selected binary strings r and put the pair $(r, H(r))$ in the bin labeled with $H(r)$. The mint process is finished when each of these bins contains 4 entries. Each bin is considered as one coin. That is, each coin is a tuple (x_1, x_2, x_3, x_4) such that $H(x_1) = H(x_2) = H(x_3) = H(x_4)$.

Figure 1: Merkle hash tree



2 Bitcoin

The cryptographic currencies in the preceding section have never been used in practice. The world changed after the cryptographic currency Bitcoin was introduced in the paper [12] by a pseudonym "Satoshi Nakamoto". Since 2009, the implementation of Bitcoin has been in operation and it has been widely adopted as one of the major cryptographic currency on the market now. The cryptography behind bitcoin is quite simple. The start coinbase by Satoshi Nakamoto is a binary string w_0. In order to mine the first bitcoin BTC, one needs to find a random number r_0 such that the first two bits of $w_1 = H(w_0, r_0)$ is 00 (that is, $w_1 < 2^{|w_0|-2}$). Anyone who finds this r_0 is rewarded with a few BTCs. The next person who finds another r_1 such that the first two bits of $w_2 = H(w_1, r_1)$ is 00 will be rewarded with a few BTCs also. This process continues and new blocks w_{i+1} keep adding to the existing block chain $w_0, \cdots, w_i$. If the frequency of finding a BTC block is less than 10 minutes, the community initiates a voting process to extend the required prefix of 0s in the hash outputs. The bitcoin is a chain $w_0, w_1, \cdots, w_n$ where w_n is the current bitcoin head that everyone works on it. The bitcoin network is a peer to peer (P2P) network and all participants work on the longest chain. There is no benefit for one to work on a shorter chain since it is a waste of time and the transaction included in these chains will not be valid. The transactions of bitcoins are included in the hash inputs so that they could be verified later. Specifically, we have

$$w_{i+1} = H(w_i, \text{TR}, r_i)$$

where TR is the Merkle hash output of the transactions that one wants to include and r_i is a random number that one finds to make w_{i+1} have a certain number 0's in its prefix. The Merkle hash tree is illustrated in Figure 1.

In the bitcoin system, a user is identified by a public key and a transaction is in the format of "Alice pays x BTC to Bob". A transaction is achieved by Alice signing the message "reference number, Bob's pub key, BTC amount x" where the reference number refers to a block w_i in the current BTC chain $w_0, w_1, \cdots, w_n$ where Alice recceived at least x BTCs in a transaction with the given reference number included in w_i. For example, the block w_i includes a transaction with this given reference number showing that Alice received certain amount of BTCs. Bitcoin transactions are described using Forth-like Scripts. The scripts enable smart contract such as "the transaction will be valid two days after all three persons have signed the contract". The Forth-like Scripts is a stack based script language and was mainly used in calculators. For example, in order to compute $25 \times 10 + 50$, one needs to initialize the stack as "[top] $25, 10, *, 50, +$ [bottom]".

Though it is argued [12] that if the majority of the users are honest then the bitcoin protocol should be reliable, Eyal and Sirer [6] showed that this may not be true. In Eyal and Sirer's attack, the adversary controls 1/3 computing powers of the entire bitcoin community and will not reveal the block it mined if it leads. The other 2/3 users will waste their time on a chain that will be abandoned at some time when the adversary reveals its own leading chain. Since users could choose arbitrary public keys for bitcoins, it is claimed in [12] that user privacy is preserved in bitcoins at certain degree. There have been significant efforts to analyze the privacy issues in bitcoin systems. Reid an Harrigan [14] defined transaction network graphs and user network graphs to analyze the coin flows in bitcoin networks. In particular, the authors in [14] proposed techniques to collape several public keys to one user node if these public keys are used as the input for a single transaction. The authors in [14] also proposed to map users to IP address based on Kaminsky [8]. Since then, transaction graphs have been used by Ron and Shamir [16] and Fleder et al. [7] to

analyze the bitcoin transaction chain. In particular, the authors [7] used Google's page rank algorithm to identify relatively "important" data from transaction networks. Furthermore, Spagnuolo[17] designed the BitIodine tool to label and cluster users in user network graphs. In particular, Spagnuolo[17] designed some web crawler to map BTC addresses to real world users and used tag data from [2] to obtain BTC addresses for gambling, online wallet, mining pool, and BTC addresses under seizure. These information were used for user clustering in user network graphs and the resulting graphs were used to analyze bitcoin transactions related to Silk Road. CryptoLoker were then analyzed based on these results. Meiklejohn et al [9] used marked coins to trace certain BTCs. That is, the authors made small transactions with many merchants to get BTC keys for these merchants. These keys are then used to cluster the user nodes in user network graphs. Meiklejohn et al [9] also used a second heuristic to cluster user nodes: change address or shadow address. In particular, the authors assume that if the output bitcoin key in a transaction has never been used before, this should be a shadow (or change address) belonging to the input user. However, this heuristic is not effective in practice since most merchants provide a newly generated bitcoin key for each transaction. There have been many proposals on privacy preserving solutions in bitcoin networks. Androulaki et al [1] tried to give a privacy definition in bitcoin networks based on traditional privacy definition in computer networks. Based on these definitions, Androulaki et al [1] implemented a simulated bitcoin network and observed that 40% user profile could be identified in the simulated environments. Ober et al [13] analyzed some global properties of bitcoin networks and their impacts on user privacy. Möser [11] analyzed three mixing services for bitcoin networks: BTC Fog, BitLaundry, Shared Wallet from Blockchain.info. Möser [11] observed that among these three services, BTC Fog and Shared Wallet has good privacy protection and tainted analysis could be used to trace bitcoins in BitLaundtry due to its lower volume per day. Moore and Christin [10] analyzed forty bitcoin exchange centers and observed that the smaller the volume, the shorter lifetime of the exchange center.

3 Ethereum

Though Forth-like Scripts in the bitcoin are sufficient for designing various kinds of smart contracts, it has a limited capability. One of the underlying philosophy in Ethereum is to include a Turing-complete programming language within the blockchain system so that any kinds of smart contracts can be supported in the blockchain. Ethereum was designed as an Internet Service Platform with the goals that anybody can upload programs to the Ethereum World Computer and anybody can request that a program that has been uploaded be executed. There are mainly two new functions in Ethereum compared with bitcoin:

- Ethereum is a blockchain with a built-in Turing-complete programming language, allowing anyone to write smart contracts and decentralized applications where they can create their own arbitrary rules for ownership, transaction formats and state transition functions.
- Bitcoin only supports "proof of work" while Ethereum supports both "proof of stake" and "proof of work" where "proof of stake" calculates the weight of a node as being proportional to its currency holdings and not its computational resources.

The runtime environment for smart contracts in Ethereum is based on the Ethereum Virtual Machine (EVM). The EVM can run any operations that are created by the user using the Turing-complete Ethereum scripting language *Solidity*. An Ethereum account is a 20 bytes string with four fields: nonce, ether balance, contract code (optional), and storage (empty by default). There are two kinds of Ethereum accounts: Ethereum Externally Owned Accounts (EOAs) and contract accounts. An EOA is linked to a private key and a contract account can only be "activated" by an EOA. A contract account is governed by its internal smart contract code which is programmed to be controlled by an EOA with a certain address. A smart contract program within a contract account executes when a transaction is sent to that account. The sender of a transaction must pay for each step of the "program" that they activate. This will include both computation and memory storage costs. Users can create new contracts by deploying code to the blockchain.

4 Infeasibility of a smart Obama-Trump contract

Since blockchains use Turing-complete script languages to draft smart contracts, people has a misconception that any kind of contracts could be implemented in blockchains. Though most financial based contracts could be implemented

using Turing-complete script languages, there are challenges in implementing smart contracts with private inputs. In this section, we analyze the limit of smart contracts that could be implemented in blockchains. In particular, we show that it is theoretically impossible to implement the so-called Obama-Trump contract.

In the legal system, there are four types of classifications of contracts with various basis: formation, nature of consideration, execution and validity. On the basis of formation, there are three types of contracts: express, implied, and quasi contracts. For an express contracts, there is an expression or conversation. For an implied contract, there is no expression. For example, sitting on an airplane incurs an implied contract between the passenger and the airline. For a quasi contract, there is no contractual relations between the partners. This kind of contract is created by virtue of law. On the basis of nature of consideration, there are two types of contracts: bilateral contracts and unilateral contracts. A bilateral contract requires considerations in both directions to be moved after the contract while a unilateral contract requires considerations to be moved only in one direction after the contract. An example of a bilateral contract is that "Alice delivers goods to Bob on January 1st and Bob pays Alice on January 15th". On the basis of execution, there are two types of contracts: executed and executory contracts. In an executed contract, the performance is completed. In a executory contracts, the contractual obligations are to be performed in future. On the basis of validity, there are five types of contracts: valid, void, voidable, illegal and unenforceable contracts. A contract that is enforceable in a court of law is called a valid contract and a contract that is not enforceable in a court of law is called a void contract. For example, a contract between Alice and Bob where Bob is a minor who has no capacity to contract is a void contract. A voidable contract is contract that is deficient in only free consent. For example, the contract between Alice and Bob where Bob has forcibly made Alice involved in the contract is a voidable contract at the option of Alice. An illegal contract contains unlawful object. An unenforceable contract has not properly fulfilled legal formalities.

With the classification of these contract types, it is important to design validation systems to check the validity of smart contracts. The following is a list of validation systems that we would like to see.

- Check whether one transaction is an implied contract.
- Check whether one transaction follows a qusai contract.
- Check whether a contract is valid, void, voidable, illegal or unenforceable?

Unfortunately, it is infeasible to design efficient validation systems to carry out these tasks due to the non-decidability of the universal Turing machine halting problem.

Furthermore, not all of these contracts are feasible in blockchain smart contracts. In particular, when private inputs are involved. As an example, we show that a bilateral contract is hard to implement if the second consideration is not digital cash (e.g., ether). In April 2011, Donald Trump made the comment [18] in an interview with ABC's George Stephanopoulos: "Maybe I'm going to do the tax returns when Obama does his birth certificate... I'd love to give my tax returns. I may tie my tax returns into Obama's birth certificate". Based on this comment, we formulate the following Obama-Trump contract and show that this kind of bilateral contract is impossible to implement as a blockchain smart contract.

Obama-Trump Contract: *Donald Trump releases his tax return forms as soon as Barack Obama releases his birth certificate.*

The infeasibility of implementing Obama-Trump Contract as a blockchain smart contract can be mathematically proved using the infeasibility results in secure multi-party computations. We first review Cleve's result [5] on the limits of coin flips when half participants are faulty.

Theorem 4.1 *(Cleve [5]) If at least half of the participants is faulty, then there is no protocol to allow an asynchronous network of participants to agree on a random (unbiased) bits.*

Cleve [5] defines a 2-processor bit selection scheme as a sequence of pairs of processors $\{(A^n, B^n)\}_{n=1}^{\infty}$ with the following properties. For each n, A^n and B^n each have access to a private supply of random bits and they can communicate with each other. If the system is executed then A^n and B^n will output bits a and b respectively within a polynomial time. Assume that the system consists of $r(n)$ rounds where each round consists of the following events: A^n performs some computations and sends a message to B^n and then B^n performs some computations and sends a message to A^n. The 2-processor bit selection scheme is said to be *correct* if after the scheme is run we have $a \neq b$ with a negligible probability. The 2-processor bit selection scheme is said to be *random* if after the scheme is run we have $a = b$ and $\left|\text{Prob}[a = 0] - \frac{1}{2}\right|$ is negligible. If one of the two processors is faulty then it is unrealistic to

expect the correctness of the scheme since the faulty processor could output a bit that is independent of the scheme run. However, it is desirable that the output of the honest processor is still random. Cleve [5] defines a 2-processor bit selection scheme to be *secure* if the following holds: For each n, if one of A^n, B^n is faulty, then $\left|\text{Prob}[c=0]-\frac{1}{2}\right|$ is negligible where c is the output of the honest processor. Cleve shows that no secure 2-processor bit selection scheme exists when one of the processors is faulty.

Assume that there is a blockchain smart contract to implement the Obama-Trump contract, then we can use this smart contract as an oracle to design a 2-processor bit selection scheme Π as follows:

`Protocol` Π: For each n, A_n checks the Obama-Trump smart contract transaction on the blockchain, if the smart contract releases Obama's birth certificate, it outputs 1. Otherwise, it outputs 0. For each n, B_n also checks the smart contract transaction. If the smart contract releases Trump's tax return form, it outputs 1. Otherwise, it outputs 0.

Theorem 4.2 *Assume that the Obama-Trump smart contract is enforced, Obama is honest and Obama releases his birth certificate randomly. That is, with 50% probability, Obama releases his birth certificate. Then the 2-processor bit selection scheme* Π *is secure 2-processor bit selection scheme.*

Proof. We distinguish the following two cases.

1. Trump is honest. In this case, both A_n and B_n output the same bit. Since Obama makes a random decision to release his birth certificate or not, the shared output is random.
2. Trump is dishonest. In this case, Obama makes a random decision to release his birth certificate or not. Thus the output of A_n is random.

□

Corollary 4.3 *Obama-Trump smart contract cannot be enforced on blockchain.*

Proof. Assume that Obama-Trump smart contract can be enforced on blockchain. Using an Obama-Trump smart contract and an honest Obama oracle (note that an honest Obama oracle can be trivially implemented), Theorem 4.2 gives a secure 2-processor bit selection scheme. This contradics with Theorem 4.1. The Corollary is proved. □

5 Smart contract scenarios

The results in the preceding section show that not all contracts could be implemented as a blockchain smart contract. However, blockchain smart contracts could do better than other technologies in many practical contract scenarios where the contract execution process takes a significant amount of time. For example, the insurance claim process involves many manual operations and requires a lot of human action. Blockchain smart contracts could help to reduce these manual steps by including some measurable parameters such as earthquake magnitude within the smart contracts. When an insured event occurs, the event information is converted to smart contract input parameters and the claim process is triggered immediately.

Smart contracts can also be used in many other scenarios where a lot of paperwork and coordination are required. For example, in the trade finance, the process of Letter of Credit issuance requires tons of physical documents. As another example, in the property rental application process, the applicant needs to submit a lot of documents such as income certificates, rental credit reports, eviction history, and other related documents to the landlord. It is noted that the user may need to submit identical documents to both trade finance vendor and landlord at different times if the user will be involved in both processes. Thus it is preferred for a user to keep all these documents in a central blockchain account and only submit appropriate reference numbers to the documents for each application. The system should be designed in such a way that the user only needs to disclose minimal mandatory information to each vendor for a specific application. For example, for a user to apply for an apartment, the system should only disclose user income, rental credit reports, and eviction reports to the landlord. While the system should not disclose user eviction reports to the trade finance organization.

Since information stored in the blockchain is publicly accessible, it is necessary to encrypt user documents in the user account. We may assume that each document in a user profile has been certified by a related agency which is also a user account in the blockchain. As an example, the user Alice's master profile may look like this:

$$\text{Alice Profile: } \text{DOC}_1, \text{DOC}_2, \cdots$$

where each document DOC_i is in the following format:

$$\mathrm{DOC}_i = \mathtt{S.Enc}_{\mathtt{K}}(F, \mathtt{Sign}_{\mathtt{Agency.pk}}(F)), \mathtt{P.Enc}_{\mathtt{Alice.pk}}(\mathtt{K}), \mathtt{Agency.pk}$$

where the document F is certified by the agency with a digital signature $\mathtt{Sign}_{\mathtt{Agency.pk}}(F)$ using the agency public key Agency.pk. The certified document $(F, \mathtt{Sign}_{\mathtt{Agency.pk}}(F))$ is then encrypted using a symmetric encryption scheme $\mathtt{S.Enc}_{\mathtt{K}}(\cdot)$ with a key K. The symmetric key K is encrypted using a public encryption scheme $\mathtt{P.Enc}_{\mathtt{Alice.pk}}(\cdot)$ with Alice's public key Alice.pk. In order for Alice to disclose the certified document $(F, \mathtt{Sign}_{\mathtt{Agency.pk}}(F))$ to the landlord, Alice needs to provide the document reference number DOC_i and the symmetric key K to the landlord.

6 Other sophisticated smart contracts

A blockchain smart contract is generally written using a blockchain scripting language such as Solidity. Thus the algorithms within the smart contract are available for public review. In some applications such as the insurance industry, the vendor may not want the public to learn the claim procecssing algorithms within the smart contract. Software obfuscation techniques may be used by smart contracts to hide these algorithms. Indeed, it is preferred to use re-usable garble circuit techniques or fully homomorphic encryption (FHE) techniques to write smart contracts in these scenarios. However, there are challenges in employing garbled circuits or FHE techniques in these scenarios since it difficult to convert plaintext inputs into garbled inputs for garbled circuits or into encrypted inputs for FHE schemes.

Public key infrastructure (PKI) is the core component for the secure Internet infrastructure. It is noted that a PKI system based on blockchain smart contract systems may be established to replace the current certificate authority (CA) based PKI systems for Internet infrastructure. It depends on the corresponding cost and security characteristics for one to consider whether to use the current CA based PKI system or to use blockchain based PKI system for the Internet infrastructure.

References

[1] E. Androulaki, G.O. Karame, M. Roeschlin, T. Scherer, and S. Capkun. Evaluating user privacy in bitcoin. In *International Conference on Financial Cryptography and Data Security*, pages 34–51. Springer, 2013.

[2] Blockchain. Address tags. `https://blockchain.info/tags`.

[3] D. Chaum. Blind signatures for untraceable payments. In *Proc. CRYPTO*, pages 199–203. Springer, 1983.

[4] D. Chaum, A. Fiat, and M. Naor. Untraceable electronic cash. In *Proc. CRYPTO*, pages 319–327. Springer-Verlag New York, Inc., 1990.

[5] R. Cleve. Limits on the security of coin flips when half the processors are faulty. In *Proc. 18th ACM STOC*, pages 364–369. ACM, 1986.

[6] I. Eyal and E.G. Sirer. Majority is not enough: Bitcoin mining is vulnerable. In *International conference on financial cryptography and data security*, pages 436–454. Springer, 2014.

[7] M. Fleder, M.S. Kester, and S. Pillai. Bitcoin transaction graph analysis. *arXiv preprint arXiv:1502.01657*, 2015.

[8] D. Kaminsky. Black Ops of TCP/IP 2011. *Black Hat USA*, page 44, 2011.

[9] S. Meiklejohn, M. Pomarole, G. Jordan, K. Levchenko, D. McCoy, G.M. Voelker, and S. Savage. A fistful of bitcoins: characterizing payments among men with no names. In *Proceedings of the 2013 conference on Internet measurement conference*, pages 127–140. ACM, 2013.

[10] T. Moore and N. Christin. Beware the middleman: Empirical analysis of bitcoin-exchange risk. In *International Conference on Financial Cryptography and Data Security*, pages 25–33. Springer, 2013.

[11] M. Moser, R. Bohme, and D. Breuker. An inquiry into money laundering tools in the bitcoin ecosystem. In *eCrime Researchers Summit (eCRS), 2013*, pages 1–14. IEEE, 2013.

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

[13] M. Ober, S. Katzenbeisser, and K. Hamacher. Structure and anonymity of the bitcoin transaction graph. *Future internet*, 5(2):237–250, 2013.

[14] F. Reid and M. Harrigan. An analysis of anonymity in the bitcoin system. In *Privacy, Security, Risk and Trust (PASSAT) and IEEE Third Inernational Conference on Social Computing (SocialCom)*, pages 1318–1326. IEEE, 2011.

[15] R.L. Rivest and A. Shamir. PayWord and MicroMint: Two simple micropayment schemes. In *International Workshop on Security Protocols*, pages 69–87. Springer, 1996.

[16] D. Ron and A. Shamir. Quantitative analysis of the full bitcoin transaction graph. In *International Conference on Financial Cryptography and Data Security*, pages 6–24. Springer, 2013.

[17] M. Spagnuolo, F. Maggi, and S. Zanero. Bitiodine: Extracting intelligence from the bitcoin network. In *International Conference on Financial Cryptography and Data Security*, pages 457–468. Springer, 2014.

[18] Donald Trump. I will release my tax returns when Obama releases his birth certificate. `http://www.businessinsider.com/donald-trump-tax-returns-obama-birth-certificate-2011-4`, 2011.