CryptoNote v 1.0

Nicolas van Saberhagen

December 12 2012

1 Introduction

Bitcoin [1] has been a successful implementation of the p2p electronic cash concept. Both professionals and the general public have come to appreciate the convenient combination of public transactions and proof-of-work as a trust model. Today, the user base of electronic cash is growing at a steady pace; customers are attracted to low fees and the anonymity provided by electronic cash and merchants value its predicted and decentralized emission. Bitcoin has effectively proved that electronic cash can be as simple as paper money and as convenient as credit cards.

Unfortunately, Bitcoin suffers from several deficiencies. For example, the system's distributed nature is inflexible, preventing the implementation of new features until almost all of the network's users update their clients. Some critical flaws that cannot be fixed rapidly deter Bitcoin's widespread propagation. In such inflexible models, it is more efficient to roll-out a new project rather than perpetually fix the original one.

In this paper, we study and propose solutions to the main deficiencies of Bitcoin. We believe that a system taking into account the solutions we propose will lead to a healthy competition among different electronic cash systems. We also propose our own electronic cash, "CryptoNote", a name emphasizing the next breakthrough in electronic cash.

2 Bitcoin drawbacks

2.1 Traceability

Privacy and anonymity are the most important aspects of electronic cash. Peer-to-peer payments seek to be concealed from third party's view, a distinct difference when compared with traditional banking. The benefits are clear: companies do not want to reveal their internal transactions and ordinary people disagree to provide any information about their personal expenses.

In particular, T. Okamoto and K. Ohta described six criteria of an ideal electronic cash, which included "privacy: the relationship between the user and his purchases must be untraceable" [2]. We define the properties of a fully anonymous payments system as follows:

Untraceability: for each incoming transaction all possible senders are equiprobable.

Unlinkability: for any two outgoing transactions it is impossible to prove they were sent to the same person.

Unfortunately Bitcoin does not meet the first criteria. Since all the transactions that take place between the network's participants are public, any transaction can be unambiguously traced to a unique origin and final recipient. Even if two participants exchange funds in an indirect way, a properly engineered path-finding method (e.g. "A star" algorithm [6]) will reveal the origin and final recipient.

Also Bitcoin doesn't seem to satisfy the second property. Some researchers stated ([3, 4, 5]) that a careful blockchain analysis may reveal a connection between users and their transactions. Although a number of the methods are disputed [7], a lot of prima facie hidden personal information can be extracted from the public database.

Bitcoin's failure to satisfy the two properties outlined above leads us to conclude that it is not an anonymous but a pseudo-anonymous electronic cash system. In comparison with the classic "chaumian" scheme with blind signatures [8] it does not provide the same level of privacy. Some solutions were proposed: ranging from "mixing services" [9] to distributed methods [10]. Both solutions are based on the idea of mixing several public transactions and sending them through some intermediary address; which in turn suffers the drawback of requiring a trusted third party.

2.2 Proof-of-work function

Bitcoin's creator Satoshi Nakamoto described the majority decision making algorithm as "one-CPU-one-vote" and used a CPU-bound pricing function (SHA-256) for his proof-of-work scheme. Since users vote for the single history of transactions order, the reasonableness and consistency of this process are critical conditions for the whole system.

First, the security issue: the network is out of danger when 51% of the mining power is under the control of honest users. Second, the progress of the system is limited: the next version of the main protocols will be applied if and only if the overwhelming majority of users supports the changes [11]. Finally the same voting mechanism is also used for collective polls about the implementation of some features [13].

This permits us to conjecture the properties that must be satisfied by a proof-of-work pricing function. Such a function must not enable a network participant to have a significant advantage over another participant; it requires a parity between common hardware and high cost custom devices. And as we can see [12], the Bitcoin pricing function SHA-256 does not have this feature: a typical video card is more effective than a CPU and ASIC devices are more powerful than GPUs.

Therefore, Bitcoin creates favourable conditions for a large gap between the voting power of participants as it violates the "one-CPU-one-vote" principle since GPU and ASIC owners posses a much larger voting power when compared with CPU owners. It is a classical example of the Pareto principle where 20% of a system's participants control more than 80% of the votes.

2.3 Other shortcomings

Irregular Emission

Bitcoin has a predetermined emission rate: each solved block produces a fixed amount of coins. Approximately every four years this reward is halved. The original intention was to create a limited smooth emission with exponential decay, but in fact we have a piecewise linear emission function whose breakpoints may cause problems to the Bitcoin infrastructure.

When the breakpoint occurs, miners start to receive only half of the value of their previous reward. The absolute difference between 12.5 and 6.25 BTC (projected for the year 2020) may seem tolerable. However, when examining the 50 to 25 BTC drop that took place recently we saw that it felt inappropriate for a significant number of members of the mining community. This event could have been the perfect moment for the malevolent individual described in the proof-of-work function section to carry-out a double spending attack [14].

Bulky scripts

The script system in Bitcoin is too complicated and heavy. It *potentially* allows to create sophisticated transactions [15], but some of its features are disabled for security reasons or have never been used [16]. Meanwhile the script (including both sender and receiver parts) for the most popular transaction in Bitcoin looks as follows:
`<sig> <pubKey> OP_DUP OP_HASH160 <pubKeyHash> OP_EQUALVERIFY OP_CHECKSIG`.
It is 164 bytes long whereas its only purpose is to check if the receiver possesses the secret key, i.e. to verify his signature.

3 Untraceable Transactions

In this section we propose a scheme for fully anonymous transactions satisfying both untraceability and unlinkability conditions. An important feature of our solution is its autonomy: the sender is not required to cooperate with other users or a trusted third party to make his transactions; hence each participant produces a cover traffic independently

3.1 Related works

Our scheme relies on a cryptographic primitive called *group signature.* Invented by D. Chaum and E. van Heyst [17], it allows a user to sign a message on behalf of a group. After signing, the user provides (for verifying) not his own single public key, but the keys of all the users of his group. A verifier is convinced that the real signer is a member of this set, but doesn't know his exact identity.

The original protocol included a Trusted Third Party (Group Manager), and he was the only one who could trace the signer. The next version — *ring signature*, introduced by Rivest et al. in [18], — was an autonomous scheme without the Group Manager and anonymity revocation. Various powerful modifications appeared later: *linkable ring signature* [19, 20, 21] allowed to determine if two signatures were produced by the same group member, *traceable ring signature* [22, 23] limited excessive anonymity by providing the possibility to trace the signer of two messages with respect to the same meta-information ("tag" in terms of [22]).

A similar cryptographic construction is also known as *ad-hoc group signature* [24, 25]. It emphasizes the arbitrary group formation, whereas group/ring signature schemes rather imply a fixed set of members.

For the most part, our solution is based on the work "Traceable ring signature" by E. Fujisaki and K. Suzuki [22]. In order to distinguish the original algorithm and our modification we will call the latter *one-time ring signature*, stressing the user's capability to produce only one valid signature under the private key. We weakened the traceability property and kept the linkability only to provide one-timeness: the public key may appear in many foreign verifying sets and the private key can be used for generating a unique anonymous signature. In case of a double spend attempt these two signatures will be linked together, but revealing the signer is not necessary for our purposes.

3.2 Definitions

Elliptic curve parametres

As a base signature algorithm we use the modern extremely fast scheme EdDSA, which was developed and implemented by D.J. Bernstein et al. [26].

Common domain parameters are:

q: prime number;

d: element of $\mathbb{F}_q$;

E: elliptic curve equation;

G: base point;

l: prime order of the base point;

$\mathcal{H}_s$: cryptographic hash function $\{0,1\}^* \to \mathbb{F}_q$;

$\mathcal{H}_p$: deterministic hash function $E(\mathbb{F}_q) \to E(\mathbb{F}_q)$.

Terms

Enhanced privacy requires some new terms which should not be confused with Bitcoin entities.

private ec-key is a standard elliptic curve secret key: a number $a \in [1, l-1]$;

public ec-key is a standard elliptic curve public key: a point $A = aG$;

one-time keypair is a pair of private and public ec-keys;

private user key is a pair (a, b) of two different private ec-keys;

tracking key is a pair (a, B) of private and public ec-key (where $B = bG$ and $a \neq b$);

public user key is a pair (A, B) of two public ec-keys derived from (a, b);

standard address is a representation of a public user key by a human-typable string with error correction;

The general transaction structure remained almost identitcal to Bitcoin's: every user can choose several independent incoming payments (transaction outputs), sign them with the corresponding private keys and send them to different destinations.

Contrary to Bitcoin's model, where a user possesses both the unique private and public keys, in the proposed model a sender generates a one-time public key based on the recipient's address and some random data. In this sense, an incoming transaction for the same recipient is sent to a one-time public key (not directly to a unique address) and only the recipient can recover the corresponding private part to redeem his funds (using his unique private key). The recipient can spend the funds using a ring signature, keeping his ownership and actual spending anonymous. The details of the protocol are explained in the next subsections.

3.3 Unlinkable payments

Classic Bitcoin addresses, once published, becomes unambiguous identifiers for every incoming payment, linking them together and tying them to the recepient. We propose a solution allowing the user to publish **a single address** and receive unconditional unlinkable payments. The destination of each output (by default) is an unique public key, derived from the recipient's address and the sender injecting random data.

First the sender performs the Diffie-Hellman exchange protocol to get a shared secret from his data and a half of the address. Then he computes a one-time destination key, using these secrets and the second half. Two different recipient ec-keys are needed for these two steps, so the standard CryptoNote address is nearly twice as large compared to a Bitcoin address. The receiver also performs the Diffie-Hellman protocol and then recovers corresponding secret key.

A standard transaction sequence goes as follows:

1. Alice wants to send a payment to Bob, who published his standard address. She unpacks it and gets Bob's public user key (A, B);

2. Then Alice generates a random $r \in [1, l-1]$ and computes the public one-time key $P = \mathcal{H}_s(rA)G + B$;

3. Alice uses P as a destination key for the output and also puts the value $R = rG$ (as part of the Diffie-Hellman protocol) somewhere into the transaction. Note that she can create other outputs with unique public keys: different recipients' keys (A_i, B_i) imply different P_i's even with the same r;

4. Bob checks every passing transaction with his private key (a, b), computing $P' = \mathcal{H}_s(aR)G + B$. If Alice's transaction was present, then $aR = arG = rA$ and $P' = P$.

5. Now Bob can recover the corresponding one-time private key: $x = \mathcal{H}_s(aR) + b$, so as $P = xG$. He can spend this output at any time by signing the transaction with x.

As a result Bob gets incoming payments, associated with one-time public keys which are **unlinkable** for a wingside spectator.

3.4 One-time ring signature

A protocol based on one-time ring signatures allows users to achieve unconditional unlinkability. Unfortunately, ordinary types of cryptographic signatures permit to trace transactions to their respective senders and receivers. Our solution to this defficiency lies in using a different signature type than those currently used in electronic cash systems.

The one-time ring signature consists of four algorithms (**GEN**, **SIG**, **VER**, **LNK**):

GEN takes public parameters and outputs an ec-pair (P, x) and a public key I.

SIG takes a message m, a set $\mathcal{S}'$ of public keys $\{P_i\}_{i \neq s}$, the pairs (P_s, x_s) and outputs a signature σ and a set $\mathcal{S} = S' \cup \{P_s\}$.

VER takes a message m, a set $\mathcal{S}$, signature σ and outputs "true" or "false".

LNK takes a set $\mathcal{I} = \{I_i\}$, a signature σ and outputs "linked" or "indep".

The general purpose of the protocol is as follows: a user produces a signature which can be checked not by a single public key, but a set of keys. The real signer is indistinguishable from the other key owners until he produces the second signature under the same keypair.

GEN: The signer picks up a random secret key $x \in [1, l-1]$ and computes the corresponding public key $P = xG$. Additionally he computes another public key $I = x\mathcal{H}_p(P)$ which we will call "key image".

SIG: The signer generates a one-time ring signature with a non-interactive zero-knowledge proof using the technique from [27]. He selects a random subset $\mathcal{S}'$ of $n-1$ other users' public keys P_i, his own keypair (x, P) and the key image I. Let $1 \leq s \leq n$ be signer's secret index in $\mathcal{S}$ (so that his key is P_s).

He picks a random element from$\{q_i \mid i = 1 \ldots n\}$ and $\{w_i \mid i = 1 \ldots n, i \neq s\}$ from $(1 \ldots l)$ and makes the following *commitments*:

$$L_i = \begin{cases} q_i G, & \text{if } i = s \\ q_i G + w_i P_i, & \text{if } i \neq s \end{cases}$$

$$R_i = \begin{cases} q_i \mathcal{H}_p(P_i), & \text{if } i = s \\ q_i \mathcal{H}_p(P_i) + w_i I, & \text{if } i \neq s \end{cases}$$

The next step is getting the non-interactive *challenge*:

$$c = \mathcal{H}_s(m, L_1, \ldots, L_n, R_1, \ldots, R_n)$$

Finally the signer computes the *response*:

$$c_i = \begin{cases} w_i, & \text{if } i \neq s \\ c - \sum_{i=1}^{n} c_i \mod l, & \text{if } i = s \end{cases}$$

$$r_i = \begin{cases} q_i, & \text{if } i \neq s \\ q_s - c_s x \mod l, & \text{if } i = s \end{cases}$$

The resulting signature is $\sigma = (I, c_1, \ldots, c_n, r_1, \ldots, r_n)$.

VER: The verifier checks the signature, reconstructing the commitments:

$$\begin{cases} L'_i = r_i G + c_i P_i \\ R'_i = r_i \mathcal{H}_p(P_i) + c_i I \end{cases}$$

Then verifier checks if $\sum_{i=1}^{n} c_i \overset{?}{=} \mathcal{H}_s(m, L'_1, \ldots, L'_n, R'_1, \ldots, R'_n) \mod l$

If this equality holds, the verifier runs the algorithm **LNK**, otherwise he rejects the signature.

LNK: The verifier checks if I has been used in past signatures (these values are stored in the set $\mathcal{I}$). Double use means that two signatures were produced under the same secret key.

The mechanizm of the protocol: using L-commitments the signer proves that he knows such x that at least one $P_i = xG$. To make this proof non-repeatable we introduce the key image as $I = x\mathcal{H}_p(P)$. The signer uses the same coefficients (r_i, c_i) to prove almost the same: he knows such x that at least one $\mathcal{H}_p(P_i) = I \cdot x^{-1}$.

If the mapping $x \to I$ is a one-way injection:

1. Nobody can recover public key from the key image and identify the Signer;

2. The signer cannot make two signatures with different I's and the same x.

3.5 Summary

With a one-time ring signature Bob can effectively hide Alice's output (i.e. his input) among others: all possible spenders will be equiprobable, even the Alice has no more information than any observer. When making a transaction Bob specifies $n-1$ foreign outputs with the same amount as his, mixing all of them without the participation of other users. Bob (as well as anybody else) does not know if any of these outputs have been spent: an output can be used in thousands of signatures as an ambiguity factor and never as a target of hiding. The double spend check occurs in the **LNK** phase when looking up in the used key images set.

Bob can choose the ambiguity degree on his own: $n = 2$ means that he will have spent the output with 50% probability, $n = 100$ gives 1%. The size of the resulting signature is linear $O(n)$, so the improved anonymity costs to Bob a bigger transaction size and higher fees. He also can set $n = 1$ and make his ring signature to consist of only one element: this will instantly reveal him as a spender.

Combining both methods (one-time tx-keys and one-time ring signatures) Bob achieves a new level of privacy in comparison with the original Bitcoin scheme. It requires him only to store one private key (a, b), and to generate a public key (A, B) to start receiving and sending anonymous transactions. For every output Bob recovers unique tx-keypairs (p_i, P_i) which are unlinkable with each other or his public key. He can spend any of them, signing each input with an untaceable ring signature.

4 Egalitarian Proof-of-work

In this section we propose and implement the new proof-of-work algorithm. Our primary goal is to close the gap between CPU (majority) and GPU/FPGA/ASIC (minority) miners. It is appropriate for some users to have a certain advantage over others, but their investments should grow at least linearly with the power. More generally, producing special-purpose devices has to be the least profitable possible.

4.1 Related works

The original Bitcoin proof-of-work protocol uses the CPU-intensive pricing function SHA-256. It mainly consists of basic logical operators and relies solely on the computational speed of the processor, therefore is perfectly suitable for multicore/conveyer implementation. However, modern computers are not limited by the number of operations per second alone, but also by memory size. While some processors can be substantially faster than others [12], memory sizes are less likely to vary between machines.

Memory-bound price functions were first introduced by Abadi et al and were defined as "functions whose computation time is dominated by the time spent accessing memory" [28]. The main idea is to construct an algorithm allocating a large block of data ("scratchpad") within memory that can be accessed relatively slowly (for example, RAM) and "accessing an unpredictable sequence of locations" in it. A block should be large enough to make preserving the data more advantageous than recomputing it for each access. The algorithm should also prevent internal parallelism, hence N simultaneous threads should require N times more memory at once.

Dwork et al [29] investigated and formalized this approach leading them to suggest another variant of the pricing function: "Mbound". One more work belongs to F. Coelho [30], who proposed the most effective solution: "Hokkaido".

To our knowledge the last work based on the idea of pseudo-random searches in a big array is the algorithm known as "scrypt" by C. Percival [31]. Unlike the previous functions it focuses on key derivation, and not proof-of-work systems. Despite this fact scrypt can serve our purpose: it works well as a pricing function in the partial hash conversion problem such as SHA-256 in Bitcoin.

By now scrypt has already been applied in Litecoin [32]. But its implementation is not truly memory-bound: the ratio "memory access time / overall time" is not large enough because each instance uses only 128 KB. This permits GPU miners to be roughly 10 times more effective and continues to leave the possibility of creating relatively cheap but highly-effcient mining devices.

4.2 Our algorithm

We propose a new memory-bound algorithm for the proof-of-work price function. It relies on random access to a slow memory and emphasizes latency dependence. As opposed to scrypt every new block (64 bytes in length) depends on the all the previous blocks, not only one, so the trade-off between memory size and CPU speed becomes exponential.

Our algorithm requires about 2 Mb per instance for the following reasons:

1. It fits in the L3 cache (per core) of modern processors, which will come into mainstream in a few years;

2. A megabyte of internal memory is almost an unacceptable size for the modern ASIC pipeline;

3. GPUs may run hundreds of concurrent instances, but they are limited in other ways: GDDR5 memory is slower than CPU L3 cache and remarkable for its bandwidth, not random access speed.

4. Significant expansion of the scratchpad would require an increase in iterations, which in turn implies overall time increases. “Heavy” calls in a trustless p2p network may lead to serious vulnerabilities, because nodes are obliged to check every new block’s proof-of-work. If a node spends a considerable amount of time on each hash evaluation, he can be easily DDoSed by a flood of fake objects with arbitrary work data (nonce values).

5 Further advantages

5.1 Smooth emission

The upper bound for the overall amount of CryptoNote digital coins is also digital: $\texttt{MSupply} = 2^{64} - 1$ atomic units. This is natural restriction based only on implementation limits, not on intuition such as "N coins ought to be enough for anybody".

To ensure the smoothness of the emission process we use the following formula for block rewards:

$$\mathit{BaseReward} = (\texttt{MSupply} - A) \gg 18,$$

where A is the amount of previously generated coins.

Difficulty

CryptoNote contains a targeting algorithm which changes the difficulty of every block. This improves the system's reaction time when the network hashrate is intensely growing or shrinking, preserving a constant block rate. The original Bitcoin method calculates the ratio of actual and target difficulty between the last 2016 blocks and uses it as the multiplier for the current difficulty. Obviously this is unsuitable for rapid recalculations (because of large inertia) and results in oscillations.

The general idea behind our algorithm is to sum all the work completed by the nodes and divide it by the time they have spent to complete the work. The measure of work are the corresponding difficulty values in each block. But due to inaccurate and untrusted timestamps we cannot determine the exact time interval between blocks. A user can shift his timestamp into the future and the next time intervals might be improbably small or even negative. Presumably there will be few incidents of this kind, we can just sort the timestamps and cut-off the outliers (i.e. 20%). The range of the remaining values is the time which was spent for 80% of the corresponding blocks.

Block size limit

Users pay others for storing the blockchain and shall be entitled to vote for its size. Every miner deals with the trade-off between balancing the costs and profit from the fees, hence sets his own "soft-limit" for creating blocks. Also the core rule for the maximum block size is necessary for preventing the blockchain from being flooded with bogus transactions, however this value should not be hard-coded.

Let M_N be the median value of the last N blocks sizes. Then the "hard-limit" for the size of accepting blocks is $2 \cdot M_N$. It averts the blockchain from bloating but still allows the limit to slowly grow with time if necessary.

6 Conclusion

We have investigated the major flaws in Bitcoin and proposed the corresponding solutions. These advantageous features and our active ongoing development make the new e-cash system CryptoNote a serious rival to Bitcoin, outclassing all its forks. We believe that a competition between concurrent currencies will be great for users, as the creators will be interested in further development of their products, adding new features, enlarging the communities and fixing bugs.

We do not consider CryptoNote as a full replacement to Bitcoin. On the contrary, having two (or more) strong and convenient currencies is better than having only one. Running two or more different projects in parallel is the natural path of digital cash economics.

Nicolas van Saberhagen

Digitally signed by Nicolas van Saberhagen
DN: cn=Nicolas van Saberhagen, email=nvsaberhagen@gmail.com
Date: 2012.12.12 15:51:58 +01'00'

A Security

Now we give a sketch of the security proof for our one-time ring signature scheme. At some points it coincides with the parts of the proof given in [22], but we are going to rewrite them with a reference rather than force a reader to rush from one paper to another.

These are the properties to be established:

- **Linkability.** Given all the secret keys $\{x_i\}_{i=1}^n$ for a set $\mathcal{S}$ it is impossible to produce $n+1$ valid signatures $\sigma_1, \sigma_2, \dots, \sigma_{n+1}$, such that all of them pass **LNK** phase (i.e. with $n+1$ different key images I_i). This property implies the double spend proof in the context of CryptoNote.

- **Exculpability.** Given a set $\mathcal{S}$, at most $n-1$ corresponding private keys x_i (excluding $i = j$) and the image I_j of the key x_j it is impossible to produce a valid signature σ with I_j. This property implies the theft proof in the context of CryptoNote.

- **Unforgeability.** Given only a set of public keys, $\mathcal{S}$, it is impossible to produce a valid signature σ.

- **Anonymity.** Given a signature σ and a corresponding set $\mathcal{S}$ it is impossible to determine the secret index j of the signer with a probability $p > \frac{1}{n}$.

B Notes on the hash function $\mathcal{H}_p$

We defined $\mathcal{H}_p$ as a deterministic hash function $E(\mathbb{F}_q) \to E(\mathbb{F}_q)$. None of the proofs demand $\mathcal{H}_p$ to be an ideal cryptographic hash function. Its main purpose is to get a pseudorandom base for image key $I = x\mathcal{H}_p(xG)$ in some determined way.

With a fixed base ($I = xG_2$) the following scenario is possible:

1. Alice sends two standard transactions to Bob, generating one-time tx-keys: $P_2 = \mathcal{H}_s(r_1 A)G + B$ and $P_1 = \mathcal{H}_s(r_2 A)G + B$.

2. Bob recovers the corresponding one-time private tx-keys x_1 and x_2 and spends the outputs with valid signatures and images keys $I_1 = x_1 G_2$ and $I_2 = x_2 G_2$.

3. Now Alice can link these signatures by checking the equality $I_1 - I_2 \overset{?}{=} (\mathcal{H}_s(r_1 A) - \mathcal{H}_s(r_2 A))G_2$.

This is possible because Alice knows the linear correlation between the public keys P_1 and P_2 and in case of fixed base G_2 she also gets the same correlation between key images I_1 and I_2. Replacing G_2 with a function $\mathcal{H}_p$, which does not preserved linearity, fixes that flaw.

References

[1] http://bitcoin.org

[2] Tatsuaki Okamoto, Kazuo Ohta, "Universal Electronic Cash," Advances in Cryptology — CRYPTO '91, LNCS 576, pp. 324–337, 1991.

[3] Fergal Reid, Martin Harrigan, "An Analysis of Anonymity in the Bitcoin System", 2012.

[4] Kay Hamacher, Stefan Katzenbeisser, "Bitcoin - An Analysis", 2011, http://www.youtube.com/watch?v=hlWyTqL1hFA

[5] Dorit Ron, Adi Shamir, "Quantitative Analysis of the Full Bitcoin Transaction Graph", 2012.

[6] Hart, P. E.; Nilsson, N. J.; Raphael, B. (1968). "A Formal Basis for the Heuristic Determination of Minimum Cost Paths". IEEE Transactions on Systems Science and Cybernetics SSC4 4 (2): 100–107.

[7] Jeff Garzik, "Peer review of "Quantitative Analysis of the Full Bitcoin Transaction Grap", 2012, https://gist.github.com/3901921

[8] David Chaum, Advances in Cryptology: Proceedings of CRYPTO '82, pp. 199–203, 1982

[9] https://en.bitcoin.it/wiki/Category:Mixing_Services

[10] http://blog.ezyang.com/2012/07/secure-multiparty-bitcoin-anonymization

[11] https://en.bitcoin.it/wiki/BIP_0034#Specification

[12] https://en.bitcoin.it/wiki/Mining_hardware_comparison

[13] http://blockchain.info/p2sh

[14] Meni Rosenfeld, "Analysis of hashrate-based double-spending", 2012 https://bitcoil.co.il/Doublespend.pdf

[15] https://en.bitcoin.it/wiki/Contracts

[16] https://en.bitcoin.it/wiki/Script

[17] D. Chaum and E. van Heyst (1991). "Group signatures". Advances in Cryptology — EUROCRYPT '91, volume 547 of Lecture Notes in Computer Science. pp. 257–265.

[18] Ronald L. Rivest , Adi Shamir , Yael Tauman, How to Leak a Secret, Proceedings of the 7th International Conference on the Theory and Application of Cryptology and Information Security: Advances in Cryptology, p.552-565, December 09-13, 2001

[19] Liu, J.K., Wei, V.K., Wong, D.S.: Linkable spontaneous anonymous group signature for ad hoc groups. In: Wang, H., Pieprzyk, J., Varadharajan, V. (eds.) ACISP 2004. LNCS, vol. 3108, pp. 325–335. Springer, Heidelberg (2004)

[20] Liu, J.K., Wong, D.S.: Linkable ring signatures: Security models and new schemes. In: Gervasi, O., Gavrilova, M.L., Kumar, V., Lagan a,A.,Lee,H.P.,Mun,Y., Taniar, D., Tan, C.J.K. (eds.) ICCSA 2005. LNCS, vol. 3481, pp. 614–623. Springer, Heidelberg (2005)

[21] Au, M.H., Chow, S.S.M., Susilo, W., Tsang, P.P.: Short linkable ring signatures revisited. In: Atzeni, A.S., Lioy, A. (eds.) EuroPKI 2006. LNCS, vol. 4043, pp. 101–115. Springer, Heidelberg (2006)

[22] Fujisaki, E., Suzuki, K.: Traceable ring signature. IEICE Trans. of Fund. E91-A(1), 83–93 (2008); Presented in PKC 2007, LNCS 4450

[23] Eiichiro Fujisaki, Sub-linear size traceable ring signatures without random oracles, Proceeding CT-RSA'11 Proceedings of the 11th international conference on Topics in cryptology: CT-RSA 2011, pages 393-415 Springer-Verlag Berlin, Heidelberg 2011

[24] B. Adida, S. Hohenberger, and R. L. Rivest. Ad-Hoc-Group Signatures from Hijacked Keypairs, 2005. At `http://theory.lcs.mit.edu/~rivest/publications`

[25] Qianhong Wu, Willy Susilo, Yi Mu, Fangguo Zhang. Ad hoc group signatures, Proceeding IWSEC'06 Proceedings of the 1st international conference on Security, pages 120-135 Springer-Verlag Berlin, Heidelberg 2006

[26] D. J. Bernstein, N. Duif, T. Lange, P. Schwabe, B. Yang, "High-speed high-security signatures",Journal of Cryptographic Engineering, September 2012, Volume 2, Issue 2, pp 77–89

[27] R. Cramer, I. Damgard, B. Schoenmakers, "Proofs of partial knowledge and simplifed design of witness hiding protocols", Advances in Cryptology — CRYPTO '94, volume 839 of Lecture Notes in Computer Science, pp. 174–187. Springer Verlag, 1994.

[28] M. Abadi, M. Burrows, M. Manasse, and T. Wobber, Moderately Hard, Memory-Bound Functions, manuscript paper to appear in Proceedings of the 10th Annual Network and Distributed System Security Symposium February, 2003.

[29] Cynthia Dwork, Andrew Goldberg, and Moni Naor. On memory-bound functions for fighting spam. In Advances in Cryptology — CRYPTO 2003, volume 2729 of Lecture Notes in Computer Science, pages 426–444. Springer, 2003.

[30] Fabien Coelho, Exponential Memory-Bound Functions for Proof of Work Protocols, 2005

[31] Colin Percival, Stronger Key Derivation via Sequential Memory-Hard Functions, presented at BSDCan'09, May 2009

[32] `http://litecoin.org`