CryptoNote+

Ilya Aldanov

December 24, 2018

1 Introduction

CryptoNote [1] protocol proved to be very popular among cryptocurrency startups, with many coins choosing it as their base layer. CryptoNote offers a set of unique features, such as unmatched anonymity, scalability and egalitarian proof-of-work. Overall, these features provide fungibility and decentralization required by any electronic cash system which pretends to be a currency.

Of many CryptoNote coins the most successful one is Monero [2]. It was not the first implementation of the protocol, but the first one to provide a transparent setup without murky premines, questionable motives of the creators, and intentionally crippled code. Its success is based on devotion of the community, its commitment to long-term goals and adherence to founding principles. Being idealistic displays a great practical value in cases where other people's money are involved.

However, Monero is not the only possible way to implement a sound CryptoNote-based currency. This paper presents several features which diverge from the CryptoNote mainline, but together form a basis for a cryptocurrency with the same basic principles but a different user experience than Monero. For breivity, we call these features "CryptoNote+".

2 The CryptoNote+ Technology

We propose the following features to extend the basic protocol:

Hybrid Mining A different mining scheme preventing a straightforward 51% attack.

Slow Emission An emission curve better suited for the real-world adoption.

Return Addresses Transaction-specific addresses anonymously linking transactions to their originators.

Tiny Addresses Short numerical addresses easy to remember and relay.

Shared Secret Secure identification scheme.

P2P Chat Community chat system embedded in P2P network.

2.1 Hybrid Mining

Distributed nature of cryptocurrency requires a way to reach consensus without any single entity dictating the outcome. The most popular way to achieve it is a blockchain with a proof-of-work mining, a network where each node has a chance to add a block to the tip of the chain with the probability of this proportional to the node's ability to do some work in the real world. The strength of such network relies on the assumption that no node can do more work than all others combined. If any single node consolidates more than 51% power, it can grow the chain in any desired direction and invalidate all attempts to correct the course. This became known as "51% attack".

CryptoNote proposes an egalitarian proof-of-work hashing algorithm, which greatly improves the weight of consumer-level hardware in consensus-making process. Monero further improves it with scheduled tweaks, designed to break any specialized hardware and make 51% attack uneconomic. At the same time the mining ecosystem rapidly evolves with increasing sophistication. There are exchange points where one can buy a hashing power equal to a small datacenter easily and at a reasonable price. While Monero's network at all times outperforms such uncommitted miners, other CryptoNote implementations suffer from the attacks of overwhelming power. It's not rare to see a miner performing at 1000% of a coin's network. Any benefits of the blockchain technology become irrelevant in this situation.

The popular solution employed by the smaller CryptoNote implementations is to slightly modify the hashing algorithm, making it unique in the hope that nobody but devoted miners will use it. Such solution is temporary at best, because the mining software learns to choose the algorithm on the fly, turning the uniqueness into a mere technicality. We propose a different approach, a mining system which combines elements of proof-of-work and proof-of-stake.

As in proof-of-work, each block is signed by a miner using the hash value exceeding the target difficulty. In addition to that we require the block to be signed by the miners of N old blocks mined over a period of time, like a year. Each of these N blocks is selected deterministically by hashing all the preceding signatures, so the next signatory is not known before the previous one. The miner is using its reward transaction private key from the old block to prove its identity and sign the current block. If the miner is offline or not responding in time, the signature can be replaced by a hash of the same difficulty as the first one. All the $N+1$ signatures are rewarded by $\frac{1}{N+1}$ of the block reward.

A miner which successfully mined a block will stay online, waiting for a chance to claim a reward for signing other miner's block. This makes it in some way similiar to proof-of-stake. The strength of the scheme comes from the fact that the miner will not sign a block that references incorrect previous block. An attacker has to provide additional $N \times 100\%$ hash power to overcome the network consensus, which makes the attack N times more costly. In case of a network fracture, the part with the biggest concentration of old miners will move faster.

The difficulty algorithm need not be changed, because the proposed scheme can be seen as a Monte Carlo calculation of the ratio of old miners online. The amount of proof-of-work will average to a stable level. Likewise, the reward received by a miner averages to $\frac{N+1}{N+1}$ over the full period if the miner stays online and provides its support to the network. This holds true even if N is changed during a network upgrade. As a coin becomes more popular, it can decrease N to decrease the chatter in P2P network.

2.2 Slow Emission

Despite the best efforts of cryptocurrency advocates, its real-world adoption is still negligible. Even after the general public got aware of its existence, the most popular use of cryptocurrency is speculation. This mindset was reinforced by an unprecedented growth of value of several prominent coins. It is assumed that the most profitable strategy is holding your deposit and waiting for a spectacular growth of price. The words and actions of "hodlers" waiting for the price to go "to the moon" have some resemblance of a cult. Cryptocurrency became a victim of its own success.

While the effort to educate the public is worth making, one may assume that the road to universal adoption and prudent valuation will be long and tiresome. A few years is not enough for such a novel concept to take roots. However, the emission curves of coins like Monero were designed with a quick adoption in mind. 77% of Monero emission was distributed in 3 years since the launch. While the wider audience starts to learn about Monero and its strengths, one may feel like hopelessly late to the free lunch. By the time the

real adoption starts to show up, most of the coins will be emitted. It may be good if one sees cryptocurrency as an eternal storage of value, and the early adopters as the new aristocracy, but it may not be that good from the point of view of an everyday currency.

We propose to significantly slow down the emission to better reflect the real-world adoption. Our proposed timeline of a coin:

The launch Zero emission to prevent any premine.

24 hours The emission starts slowly rising.

1 week Miners get aware of the coin.

1 year The coin becomes established with all the necessary software available.

3 years The public starts to notice the coin. Speculation leads to great volatility of price.

5 years The emission peaks. Some real-world adoption starts to show up.

7 years The emission slowly decreases, leading to a frantic activity of the miners to get the last coins.

10 years The emission switches to the tail phase. The price stabilizes, the mining becomes a marginally profitable business.

The tail emission is an important feature of the emission curve. It has two purposes. First, it gives miners the incentive to support the network. Second, it replaces the lost coins and keeps the healthy circulation. Coin loss is an inevitable process which should always be taken into account.

2.3 Return Addresses

The most important feature of CryptoNote protocol is anonymity. Anonymity is not achieved by a single silver-bullet like operation or a method, but rather every part of the protocol is designed to provide some aspect of anonymity. CryptoNote shows its strength in many possible attack scenarios. For example, if a malicious entity forces a user to concede the secret key, the information the entity obtains reveals only the user's operations and balances. The addresses of the counterparties are hidden. It's not possible to learn who sent the coins, and to whom the coins were sent, only the amounts are revealed.

While this is an important feature, it introduces some inconveniences for the users. Sometimes it's desirable to have a link back to the entity that sent you funds. Merchants may want to do a partial refund, exchanges may want to return the erroneously transferred funds, friends may want to return money they borrowed. Out-of-channel communication always has a danger of mistakes, delays or fraud. There's no easy and reliable way to obtain the address of the originator of an incoming transaction.

We propose to include a return address into the header of every transaction, which would serve as a 100% reliable link back to the transaction originator. To prevent any two transactions to be linked together by the same return address, we propose to generate such address from the transaction public key and the sender private key, making it effectively a subaddress [3].

Suppose that Alice wishes to send some of her hard-earned coins to someone. Alice's master wallet address is $(A, B) = (aG, bG)$, where a and b are secret scalars and G a common elliptic curve basepoint. Alice prepares the transaction the usual way, computing the transaction public key R. In addition, she computes the following subaddress:

$$\begin{aligned} D_R &\equiv B + H_s(aR, -1)G \\ C_R &\equiv aD_R \end{aligned}$$

Here H_s is a cryptographic scalar hash function. The return address is defined as the pair of points (C_R, D_R), which Alice includes in the transaction header. She is assumed to also have a hash table stored on her computer that maps $D_R \mapsto R$.

The index -1 is chosen to prevent collision with a change output, which may be included in the transaction. The change output public key is calculated by a similiar formula:

$$P_{\text{change}} \equiv H_s(aR, i)G + B$$

Here i is the index of the output. Choosing -1 for the return address allows to use the same calculation routines.

If anyone wants to send funds to the return address, they interpret it as a standard subaddress. All the usual subaddress sending routines are applied.

When Alice scans incoming transactions, she checks each output public key P' (with associated transaction public key R') by computing the following:

$$D' \equiv P' - H_s(aR')G$$

If Alice sees that this value D' maps to public key R in her local hash table, she is assured that the output was sent to the return address (C_R, D_R). In order to use her funds as the input to a later transaction, Alice needs to be able to determine the private key associated to P'. She can do this easily using the public key R returned from the hash table lookup:

$$p' \equiv H_s(aR') + b + H_s(aR, -1)$$

This succeeds since

$$\begin{aligned} p'G &= (H_s(aR') + b + H_s(aR, -1))G \\ &= H_s(aR')G + B + H_s(aR, -1)G \\ &= H_s(aR')G + D_R \\ &= P'. \end{aligned}$$

This routine should not be done independently, but as an extension of the existing subaddress scanning routine.

Can anyone link the return address to Alice's master address? In order to do so, an adversary should be able to recover B from the published D_R. The only way to do it is to calculate $H_s(aR, -1)$. While R is publicly known, a is a secret. At the same time $aR = arG = rA$. The adversary can guess A, but the transaction private key r is a secret.

If Alice restores the wallet from the seed, she will be able to immediately reconstruct the return address hash table. Every time she encounters an outgoing transaction, she adds $D_R \mapsto R$ to the table. No knowledge of secret keys or calculations are required.

Some of the possible use cases for the return addresses outlined below:

- Full or partial refund by a merchant.
- Return of unidentified funds. An exchange may return funds sent to an erroneous or closed account.
- Non-interactive services. Some respectable institution may offer short-term deposits, paying interest in consecutive return transactions.
- In-channel delivery of information. The return transaction may contain a password to some service in the low-denomination part of the amount.
- Broker services. The return transaction may contain an amount of something bought. A return to the return may be interpreted as a sale order.

2.4 Tiny Addresses

Every time a transaction is recorded in the blockchain, it contains a return address of the sender. Everyone can see the address and send funds to it as many times as necessary. In effect, the transaction can be seen as a business card publicly available in the blockchain. Instead of giving her main address or a subaddress, Alice can provide the hash of one of her transactions, which is twice as short as the full address. Anyone who wants to send funds to Alice needs to access the blockchain anyway, so it just adds a little overhead to resolve the transaction hash to an address. The blockchain turns into a public directory.

But we propose to go even further. Each transaction can be assigned an index, the same way each output has an index in CryptoNote implementations. The most recent transactions can see their index changed subject to reorgs, but after some height the index becomes effectively permanent. Instead of the hash, Alice can provide the index of one of her transactions as her address. We call this substitute a "tiny address".

When encoded as a decimal number (the preferred method of encoding for the worldwide adoption), a tiny address constitues a small number of digits, easy to remember and transmit verbally, or enter manually. Alice doesn't have to rely on computer hardware to store her address properly, or computer software to copy an address from a storage properly. All these intermediaries could be compromised by malicious parties, and instead of providing her address to a customer, she may unwillingly provide a third party address. The funds sent to her could be stolen. Tiny addresses solve this by allowing her to remember something like 7 digits and write them on a piece of paper in presence of a customer.

However, a few precautions apply. There should be enough confirmations of the transaction before the tiny address displayed to the user. Tiny address is just a shortcut, it has no dependence on the secret key. If it changes in a reorg, the funds sent to the previous one can never be recovered. Also, users should be aware that they link the transaction to themselves. The breach of anonimity is no more serious than receiving funds from an exchange or a mining pool, but one should be aware that the outputs of the transaction are linked. In this case one should follow the recommended best practices, like a few churns (which could be used to obtain more tiny addresses later). Also, a special kind of transactions without outputs can be considered for implementation.

Tiny addresses, once implemented, could greatly enhance the user experience. One can envision a mobile wallet application, after being installed by an inexperienced user, connecting to a remote node and asking for a grant, just enough to cover the network fee. Upon receiving the grant, it submits a zero-output transaction, waits for a hundred confirmations, and presents the user with her new tiny address. All this happens transparent to the user, with a polite notification "Please wait several hours before we generate your new address." After the setup, she can use the address to receive funds in a secure, anonymous and reliable way without any inconveniences of frightening full addresses.

2.5 Shared Secret

Payment ID field is used by many CryptoNote coins to identify incoming funds. This field proved to be a weak spot of the protocol. Its unencrypted version could be used to link transactions and track users of popular exchanges. But even the encrypted version leaks information about the nature of the transaction, since it is used mostly by exchanges and merchants. Another significant problem is that a transaction can have only one Payment ID, making multi-destination payments impossible. This is especially troublesome for mining pools, which have to make payments to miners with Payment ID in separate transactions, incurring fees and delays, since usually there are less unlocked outputs than miners.

At the same time some form of identification is necessary. Subaddresses provide some, when transmitted over a private channel. But return addresses are published in the open directory. When receiving a trans-

action to a return address, there's no way to tell if it comes from the original receiver or a third party. And since there can be a chain of return transactions, it is desirable to have a static Conversation ID linking it throughout.

We propose to include mandatory 8-byte field into each output. This field must be symmetrically encrypted using $H_s(rA, i)$ as the encryption key. In case of a subaddress the key becomes $H_s(sC, i)$. In both cases the receiver obtains the key by calculating $H_s(aR, i)$. We call the decrypted plaintext a "shared secret".

Being mandatory, shared secrets can't be used to fingerprint outputs, especially the change output. Also, they do not limit the number of outputs in a transaction. And 8 bytes should be enough for most use cases.

But we also propose to reserve a few bits of the plaintext for a data type field. Describing the type of the shared secret bit string can greatly improve user experience. Some of the possible types are:

Zero A string of zeroes.

Random A string of random bits generated by the sender.

ID Identifiaction string generated by the sender or provided by the receiver. A return transaction should relay this string.

Text 6-bit encoding can provide up to 10 latin alphanumeric characters. Can be used as a comment or a password or a command to a service (like "BUY 600", "SELL ALL").

Address A tiny address of a third party.

Top Secret A string further encrypted by $H_s(rB, i)$ (or $H_s(sG, i)$ in case of a subaddress*). The receiver must have the spend key to decrypt.

The change output's shared secret may be used to store a comment describing the transaction. The user should be made aware that such a comment is stored in the blockchain and could be used against her. On the other hand, if the wallet is used with the public viewkey, comments can explain to auditors the nature of each spending.

2.6 P2P Chat

The most valuable part of any open-source cryptocurrency is its community. The community includes the developers, who enhance and adapt the software, the advocates and envoys, who spread the knowledge and help the newcomers, and just the regular users, who give meaning and purpose to the life of the coin. All these people need a place to discuss and coordinate their actions. It may be a nontrivial task to contact the developers of a particular coin. Due to the many risks involved some of them cherish their anonymity and prefer to use obsure places and protocols to communicate. But even the well-known public places like reddit can be subject to a government action, fracturing and dispersing the community.

At the same time most people involved with the coin have a specific software designed to keep them connected to the coin's P2P network. We propose to make use of this fact and build a reliable P2P chat system, with all the features to make any third-party software and services unnecessary. Network-level anonymization technologies like Kovri could make this system extremely secure and independent. Anyone could easily reach the developers and become one of them, while keeping his identity and whereabouts hidden.

*$sG = sGd/d = sD/d = R/d$

3 Conclusion

We have presented a set of features that may not be immediately useful for an existing CryptoNote implementation like Monero. But implemented in a new coin, CryptoNote+ can offer a new flavour of cryptocurrency experience while retaining all the stregths of its base protocol.

References

[1] Nicolas van Saberhagen. Cryptonote v2.0, 2013.

[2] Monero project. `https://getmonero.org/`.

[3] Sarang Noether and Brandon Goodell. Mrl-0006: An efficient implementation of monero subaddresses, 2017.