ECLIPSE: FIRST CRYPTOGRAPHICALLY ANONYMOUS BITCOIN CORE BLOCKCHAIN

ECLIPSE DEVELOPERS

Abstract. In this paper, we describe Eclipse, an implementation of the first cryptographically anonymous decentralized currency that is based on the bitcoin codebase. Like the Cryptonote-based currency Monero [3], our decentralized currency uses ring signatures to obfuscate the sender, and stealth addresses to hide the receiver.

However, unlike other implementations that have proven to be broken, for instance Shadowcash [6, 5], we obtain the key image using the "Try-and-Increment" approach. Try-and-Increment is a cryptographically secure method for mapping a scalar hash to an EC point. The security of Try-and-Increment comes from the fact that this method does not rely on the group generator (i.e. basepoint) of the EC.

In addition, unlike ZCash [8], we do not rely on zero-knowledge proofs and the destruction of any one shard of the master key using secure multiparty computation. This protocol has the potential that the master key could be re-combined if all the shards are not destroyed and the trusted parties are in collusion. In such a case, new counterfeit tokens could be created that are indistinguishable from legitimate tokens. In contrast, ring signatures do not rely on trusted parties, and it is impossible to create a counterfeit balance.

1. Introduction

In 2009, Satoshi Nakomoto astounded the academic and cryptographic communities when he released the blockchain technology to the world, which he aptly named 'Bitcoin' [4]. This scheme was the first to combine economic incentives via block mining rewards to secure the authenticity of a distributed database using Proof-of-Work (PoW) consensus protocol. The Bitcoin protocol is necessarily pseudo-anonymous and does not provide a mechanism to obsfucate certain aspects of transactions. Eclipse, which is based on the original bitcoin codebase, uses ring cryptography to hide the sender of a transaction and stealth addresses to hide the receiver in order to obtain complete anonymity.

In 2, we provide multiple use cases for ring signatures. To make this as self-contained as possible, we review Stealth Addresses In 4.1 and Ring Signatures in 4.2, and we outline the notation that will be used to describe the mechanisms of ring cryptography in 3. In 5, we conclude with a description of the Try-And-Increment algorithm.

2. Examples

In short, ring cryptography anonymizes sender information, and stealth transactions anonymize receivers. Numerous examples exist as to the uses of ring cryptography in anonymizing sender details. We provide ones that relate to diplomacy, anonymous donations, anonymous asset transfers, and whistle-blowing.

2.1. Anonymous Proposals.

Let us consider an organization (a Homeowner's Association, Small Business Association, Federated Nations, United Nations, Intergalactic Senate, etc.), made up of any type of entity, i.e. geopolitical nation-states, businesses, individuals, etc. where some type of proposals are being put forth by the various entities.

Perhaps the United Nations is considering anonymous proposals for negotiating a trade agreement between different nations. Say that Israel supports a position that is advantageous to Israel, but is not advantageous to the United States. However, if Israel were to publicly propose such a provision, some type of political backlash would occur. A cryptographic scheme that allows for an anonymous proposal would shield the knowledge of which member nation made the proposal.

With ring signatures, every nation could be equally as likely as any other to make a proposal. Indeed, additional information could be used to determine which nation is more likely than another to try to add such a provision, but nothing could be definitively proven unless a nation chose to reveal that they were behind the proposal.

This type of anonymous proposal can be used in any type of group in which there a multitude of wide-ranging interests and an incentive from Machiavellian actors to anonymize a proposal.

2.2. **Anonymous Donations.** With the advent of meta-data collection, donating to certain political causes or even religious organizations can place a donator on a watch list. For instance, a person supports non-aggression principles and donates to causes that promote the ending of the war state. Similarly, people who donate to organizations that advocate sustainabilty, an end to police abuse, stopping the surveillance state, media watch-groups, government watch-groups, etc. are potentially all at risk of being monitored. Within the United States, the latest Supreme Court rulings indicate that all donations are regarded as free speech, but this ruling does not prohibit governmental agencies from deeming such activity as problematic. Indeed, this mindset is not isolated to the United States.

2.3. **Avoiding Oppressive Rules of Nations.** In the United States of America, campaign contribution limits exist for an individual to donate to a political candidate. The People's Republic of China has currency export laws that limit how much capital citizens or workers within the PRC can send to entities outside the PRC in any given calendar year. Since the Bitcoin public blockchain is pseudonymous, one could conceive that in the future the PRC would crack down on Bitcoin transactions as such an action could be perceived by the PRC as currency leaving its jurisdiction.

Ring signatures allow any entity (natural or fictitious person) to circumvent the contribution or export limits. Cryptotokens with properly implemented ring signature cryptographic schemes to anonymize senders are thus impervious to unjust laws that limit political speech or currency exportation.

2.4. **Anonymous Title Transfers of Property.** Any type of asset, such as artwork, vehicles, land, stock certificates, books, etc. could also be recorded on the blockchain. However, the transfer of property does not require the knowledge of the seller, only the authenticity of the asset itself. Obsfucation of the seller of a valuable piece of art, for instance, can be made possible with the use of ring signatures. In general, all property transfer can be made anonymous.

2.5. **Leaking Confidential Material.** One final application of ring signatures is the ability for high ranking government officials to anonymously leak documents to the public. By placing multiple officials' public keys in a ring, the leaker can sign a message and become an anonymous government whistle blower. Whistle blowing can occur beyond the leaking of government data, and a leaker could reveal deceptive practices within a private company, educational institution, or nonprofit.

3. Notation

We will use the following conventions:

(k_i, K_i) :	a private / public key pair;	k_i is private and K_i is public, with i being arbitrary
ℓ :	a large prime ;	$\ell = 2^{256} - 2^{32} - 2^9 - 2^8 - 2^7 - 2^6 - 2^4 - 1$
$\mathbb{F}_\ell$:	a field ;	the prime field of order ℓ
$\mathcal{E}$:	the secp256k1 elliptic curve ;	$\mathcal{E} = \{(x, y) \in \mathbb{F}_\ell \times \mathbb{F}_\ell \mid y^2 = x^3 + 7\}$
G :	a base point of $\mathcal{E}$;	G generates $\mathcal{E}$, i.e. $< G >= \mathcal{E}$
n :	the order of the base point G ;	$n = \#G$, a number slightly less than 2^{254}
H_s :	a cryptographically secure hash function;	sha-256d $\doteq H_s : \{0,1\}^* \mapsto \mathbb{Z}_{2^{256}}$
T :	Try-and-Increment ;	$T : \mathbb{Z}_{2^{256}} \mapsto \mathcal{E}$
H_p :	a deterministic hash function ;	$\mathcal{E}(\mathbb{F}_\ell) \mapsto \mathcal{E}(\mathbb{F}_\ell)$ with $H_p(K_i) \doteq k_i T(H_s(K_i))$

Notation is slightly abused with H_s taking a point on the elliptic curve as input to the function. However, one can encode the point into a bit string by taking the binary expansion of the x-coordinate of the point and an additional bit to indicate the sign of the y-coordinate. This convention is used throughout the remainder of this paper.

In addition, we use the same definitions from the CryptoNote white paper [7] which we restate here for completeness.

A **private elliptic curve key** is a standard elliptic curve private key: a number $p \in [1, n-1]$;
A **public elliptic curve key** is a standard elliptic curve public key: a point $P = pG$;
A **one-time keypair** is a pair of private and public elliptic curve keys, i.e. $(p, pG = P)$
A **private user key** is a pair (p, q) of two different private elliptic curve keys;
A **public user key** is a pair (P, Q) of two public elliptic curve keys derived from (p, q);

4. Stealth Addresses And Ring Signatures

Alice wants to send Bob Eclipse without Bob, or anyone else, knowing she is the sender. In addition, Bob wants to hide the fact that he received Eclipse from anybody. However, Alice wants to be able to prove to any party that she is the sender of Eclipse. Likewise, Bob wants the same functionality to prove that he is the receiver to any party. Anonymous sending and receiving can be accomplished with the help of *ring*[1] *signatures* and *stealth addresses*, respectively. *Stealth addresses* have a long history of use in distributed ledger technologies starting with VertCoin [1].

A *ring signature* is a cryptographic signing protocol that takes N public keys (called the *ring*[2]), a message M signed with a private key associated to one of the N public keys, and then hides the original signer. The only information that can be determined is that the signer is one of N parties, i.e. one person from the set of N public keys is the signer.

In order to obtain two-way anonymity, both the sender and the receiver must possess two public / private key pairs. Suppose Alice's two private / public key pairs are (k_1, K_1) and (k_2, K_2) and that Bob's two private / public key pairs are (k_3, K_3) and (k_4, K_4).

We adopt the CryptoNote protocol [7]. For completeness, we describe both the cryptographic schemes for stealth addresses (originating with VertCoin [1]) and ring signatures.

4.1. Anonymous Recipients.

A standard stealth transaction works in the following way.

1. Alice obtains Bob's public keys K_3 and K_4.
2. Alice generates a random $r \in \mathbb{Z}_p$ and computes a one-time public key: $Q = H_s(rK_3)G + K_4$.
3. Alice uses Q as a destination key for the output and includes $P = rG$ (in order to perform a Diffie-Helman exchange) somewhere in the transaction.
4. Alice sends the transaction.
5. Bob checks every passing transaction with his private key (k_3, k_4) and computes $Q' = H_s(k_3P)G + K_4$. If Alice's transaction that was sent to Bob is in one of them, then $k_3P = k_3rG$ and $Q' = Q$.
6. Bob generates the corresponding one-time private key : $x = H_s(k_3P) + k_4$, in order to make $Q = xG$. At any point, Bob can spend this output by signing a transaction with x.

By monitoring every transaction (Step 5), Bob obtains all incoming payments. In addition, by design, all incoming payments are associated to a one-time public key (Q in Step 2) which is unlinkable to any outside observer.

4.2. Anonymous Senders.

Obtaining a proper key image for use in the ring cryptographic scheme is necessary in order to ensure de-anonymization does not occur. Indeed, the original ShadowCoin implementation of ring signatures used the hash of the public key to generate the key image which subsequently allowed for portions of ShadowCoin blockchain to be de-anonymized. After carefully reviewing various cryptographic literature, we decided on the algorithm proposed by Icart [2] and are the first to apply this novel algorithm in our codebase to achieve complete anonymity of sender details.

Our one-time ring signature algorithm can be broken into four separate algorithms: γ, σ, ν, λ.

1. γ: Input: public parameters for the elliptic curve used and a generator G. Output: elliptic curve private-public key-pair (p, P) and another public key I.

[1]Algebraists should not become confused as this method does not use algebraic *rings*.
[2]Once more, not to be confused with an algebraic *ring*.

The signer of a message chooses a random secret key $p \in [1, n-1]$ and calculates the associated public key $P = pG$. He also computes a public key $I = p\mathcal{H}_p(P)$ which we call the **key image**.

2. σ: Input: a message m, a set $\mathcal{R}'$ of public keys $\{P_i\}_{i \neq s}$, an elliptic curve pair (P_s, p_s). Output: a signature σ and a set $\mathcal{R} = R' \cup \{P_s\}$.

   Using a non-interactive zero-knowledge proof, the signer creates a one-time ring signature.

   He selects a random subset $\mathcal{R}'$ of size z of other users' public keys P_i, his own keypair (p, P) and key image I. Let $0 \leq s \leq z$ be the signer's secret index in $\mathcal{R}$, i.e. the signer's public key is P_s.

   He then picks a random $\{q_i \mid i = 0 \ldots z\}$ and $\{w_i \mid i = 0 \ldots z, i \neq s\}$ from $\{1, \ldots, n\}$ and uses the following transformations:

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

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

   The non-interactive zero-knowledge challenge is $c = \mathcal{H}_s(m, Y_1, \ldots, Y_z, U_1, \ldots, U_z)$.

   The signer calculates the response:

$$c_i = \begin{cases} w_i, & \text{if } i \neq s \\ c - \sum_{i=0}^{z} c_i \mod n, & \text{if } i = s \end{cases}$$

$$u_i = \begin{cases} q_i, & \text{if } i \neq s \\ q_s - c_s p \mod n, & \text{if } i = s \end{cases}$$

   The returned signature is $\sigma = (I, c_0, \ldots, c_z, u_0, \ldots, u_z)$.

3. ν: Input: a message m, a set $\mathcal{S}$, and a signature σ. Output: true or false.

   A signature is verified by applying the following two inverses,

$$\begin{cases} Y_i' = u_i G + c_i P_i \\ U_i' = u_i \mathcal{H}_p(P_i) + c_i I \end{cases}$$

   and a test for equality is performed on

$$\sum_{i=0}^{z} c_i == \mathcal{H}_f(m, Y_0', \ldots, Y_z', U_0', \ldots, U_z') \mod n.$$

   If these are equal, then the next algorithm λ is called. Otherwise, the signature is rejected.

4. λ: Input: a set $\mathcal{I} = \{I_i\}$, a signature σ. Output: Linked or Independent.

   A test is conducted as to whether or not I has been used in past signatures by searching for the values in $\mathcal{I}$. If a duplicate entry in $\mathcal{I}$ exists, then two signatures were generated with the same secret key.

   Once the Y-transformations are applied, the signer can prove that he knows a p such that at least one $P_i = pG$. To prevent repeats, we introduce the key image $I = p\mathcal{H}_p(P)$. The signer uses the same coefficients (u_i, c_i) to nearly prove an equivalent statement, i.e. that he knows a p such that at least one $\mathcal{H}_p(P_i) = I \cdot p^{-1}$.

   If the map from $p \mapsto I$ is injective, then

   (1) Neither the public key from the key image nor the identity of the signer can be recovered; and
   (2) The signer cannot make two signatures with different key images I's and the same p.

5. Try-and-Increment

Try-and-Increment, based on the algorithm proposed in [2], is the algorithm we chose for generating a proper key image.

The 'Try-and-Increment' method is a function that takes as input a 256-bit hash. This 256-bit hash is then mapped to the base field, $\mathbb{F}_\ell$ by taking the remainder of the hash modulo ℓ. Let's call this field element

```
Input: $x$ - a 256-bit hash.
Output: $(x, (x^3 + 7)^{\frac{1}{2}}) \in \mathcal{E}$.
$x \to x \mod \ell$.
while $x^3 + 7$ is not a quadratic residue do
    $x \to x + 1$
end while
return P $= (x, (x^3 + 7)^{\frac{1}{2}})$
```

x. $x^3 + 7$ is then tested to see if it is a quadratic residue of $\mathbb{F}_\ell$. If it is, then the algorithm returns as a point on the secp256k1 elliptic curve $(x, (x^3 + 7)^{\frac{1}{2}})$. Otherwise, increment x by 1 and try again.

On average, the Try-and-Increment algorithm fails slightly less than half the time at each iteration. This can be readily seen by the fact that any prime field has exactly $\frac{\ell-1}{2}$ quadratic residues. Indeed, the hashing space is effectively reduced by half, due to this property. However, this can be slightly improved by returning $P = (x, -(x^3 + 7)^{\frac{1}{2}})$ when any increment occurs, although is not implemented in our scheme, nor is this improvement necessary in practice due to how long it would take for a collision to occur.

In fact, we decided to use a modified form of Try-and-Increment and its inclusion is our only modification to the optimized libsecp256k1 library that is used in `libbitcoin`. Our modification is perfectly valid and instead of incrementing by 1, we increment by i on the $i'th$ iteration, i.e. if i iterations occur, we increment by the i'th triangular number.

```
Input: $x$ - a 256-bit hash.
Output: $(x, (x^3 + 7)^{\frac{1}{2}}) \in \mathcal{E}$.
$x \to x \mod \ell$.
$i \to 1$.
while $x^3 + 7$ is not a quadratic residue do
    $x \to x + i$.
    $i \to i + 1$.
end while
return P $= (x, (x^3 + 7)^{\frac{1}{2}})$.
```

6. Conclusion

Eclipse Crypto is the first distributed ledger that properly implements the ring signature scheme outlined originally by CryptoNote [7] and uses the bitcoincore library. In addition, we used the sha-256d algorithm as a means to fairly distribute the tokens via PoW prior to switching to a PoS consensus protocol.

In this paper, we provide multiple use cases for ring signatures, give an overview of stealth addresses and ring signatures, and describe two implementations of Try-And-Increment.

References

[1] Vertcoin Developers. Vertcoin: Bitcoin Upgraded. `https://vertcoin.org/`, 2014.
[2] Thomas Icart. How to hash into elliptic curves. In *Advances in Cryptology-CRYPTO 2009*, pages 303–316. Springer, 2009.
[3] Monero Research Lab. Get Monero. `https://lab.getmonero.org/`, 2014.
[4] Satoshi Nakamoto. Bitcoin: A peer-to-peer electronic cash system, 2008.
[5] Shen Noether. Broken Crypto in Shadowcash. `https://shnoe.wordpress.com/2016/02/11/de-anonymizing-shadowcash-and-oz-coin/`, 2016.
[6] Rynomster and Tecnovert. Shadow: Zero-knowledge Anonymous Distributed Electronic Cash via Traceable Ring Signatures. `https://github.com/shadowproject/whitepapers/blob/master/shadowcash_anon.pdf`, 2015.
[7] Nicolas van Saberhagen. Cryptonote v 2. 0. `https://cryptonote.org/whitepaper.pdf`, 2014.
[8] Nathan Wilcox. How To Generate SNARK Parameters Securely. `https://z.cash/blog/snark-parameters.html`, 2016.