Stealth Address and Key Management Techniques in Blockchain Systems

Nicolas T. Courtois[1] and Rebekah Mercer[1,2]
[1]*Computer Science Department, University College London, London, U.K.*
[2]*R&D division, Clearmatics Technologies Ltd, London, U.K.*
{*n.courtois, rebekah.mercer.15*}*@ucl.ac.uk*

Keywords: Applied Cryptography, Bitcoin, DarkWallet, CryptoNote, ShadowCash, Key Management, Privacy, Anonymous Payment, Stealth Address Technique, Audit Capability, ECDSA, HD Wallets, BIP032, Leakage-resistant Cryptography.

Abstract: Bitcoin is an open source payment system with a market capitalization of about 15 G$. During the years several key management solutions have been proposed to enhance bitcoin. The common characteristic of these techniques is that they allow to derive public keys independently of the private keys, and that these keys match. In this paper we overview the historical development of such techniques, specify and compare all major variants proposed or used in practical systems. We show that such techniques can be designed based on 2 distinct ECC arithmetic properties and how to combine both. A major trend in blockchain systems is to use by Stealth Address (SA) techniques to make different payments made to the same payee unlikable. We review all known SA techniques and show that early variants are less secure. Finally we propose a new SA method which is more robust against leakage and against various attacks.

1 INTRODUCTION

Bitcoin has been in existence for nearly 8 years. It is a digital currency, payment and final clearing/settlement system and technology, a distributed property register and digital notary service, all in one. Bitcoin allows owners to authorize the transfer of their coins using digital signatures. Currently, bitcoin uses ECDSA cryptography with SHA256 and the secp256k1 curve. Cryptographic literature does not provide an answer to whether or not ECDSA is provably secure, and whether there is an attack on ECDSA other than computing discrete logs. However it is widely believed that ECDSA is secure modulo some usage precautions.

Cryptographic literature shows that both RSA and ECDSA-based public key cryptography can fail if deployed at a large scale if keys and random nonces are generated with insufficient entropy. More precisely there are three major ways in which ECDSA can fail in practice, in systems such as bitcoin: due to bad or repeated randoms (Courtois3d, 2015; N.T. Courtois, 2014), due to weak or user-chosen passwords (N.T. Courtois, 2016) and due to poor key management (Courtois3b, 2015; S. Eskandari, 2015; G. Gutoski, 2015). There are also combination attacks which exploit several of the above properties (N.T. Courtois, 2014; Courtois3d, 2015). In this paper we concentrate on the questions of private/public key management. The usage of such techniques have greatly increased in crypto currency systems in the recent years.

1.1 Why Key Management?

We refer to (N.T. Courtois, 2014; Courtois3b, 2015; S. Eskandari, 2015; G. Gutoski, 2015) for a detailed discussion on how the need for key management emerges in bitcoin and in the industry at large. Main reasons are the necessity to use several keys due to poor anonymity of existing blockchain systems, possibility to hide the public key and to use each key only once for security reasons, questions of reliable backup of bitcoin wallets and cold storage, etc. In recent works on this topic cf. (N.T. Courtois, 2014) the primary reasons to use advanced key management techniques in bitcoin were first just to diversify keys to be used in different transactions, then to develop so called "Type 2 techniques" cf. (N.T. Courtois, 2014; Courtois3b, 2015). Here the important property , which we will also need in this paper, is to have "Audit Capabilities" which allow third parties to derive public keys from certain "master public keys" and without knowledge of private keys.

More recently the Stealth Address techniques have become popular. These techniques have **an ad-**

Courtois, N. and Mercer, R.
Stealth Address and Key Management Techniques in Blockchain Systems.
DOI: 10.5220/0006270005590566
In *Proceedings of the 3rd International Conference on Information Systems Security and Privacy (ICISSP 2017)*, pages 559-566
ISBN: 978-989-758-209-7

ditional major objective: privacy for the receivers of moneys. These methods allow public keys which appear in the blockchain to be totally disconnected from "stealth" public keys which are advertised by merchants. In the same way as in HD Wallet solutions (N.T. Courtois, 2014; Courtois3b, 2015) here again, the public keys advertised serve as "master public keys" from which ephemeral public keys are derived. In this paper we study all major variants of such techniques, their security, we propose more general methods, and show how can they be made even more robust against certain attacks.

1.2 Key Management with Audit - Main Principle

We outline the primary methods used in creating "Auditable" key management sometimes called "Type 2 techniques" cf. (N.T. Courtois, 2014; Courtois3b, 2015). The crucial property we need is to be able to derive public keys independently from the private keys, or rather do both in bulk from some sort of "Master" public key[1] and a "Master" private key, cf. Fig. 1 below. We need two key Child Key Derivation (CKD) functions, a "Private CKD" function and a "Public CKD" function as shown on Fig. 1.

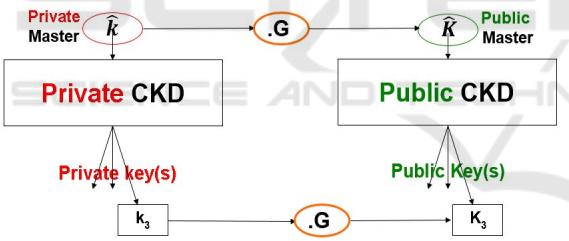

Figure 1: General key derivation principle from (N.T. Courtois, 2014) with modified notations. It must be such that the diagram commutes and public keys K_i obtained either way are identical.

The crucial property is then that the corresponding lower level keys match. This property means that our diagram in Fig. 1 should commute.

One-wayness. In (N.T. Courtois, 2014) it is also required that all the 4 arrows (derivations) are one-way functions and that the public master key/seed should NOT reveal the private master key/seed or any of the private keys.

[1] Sometimes a word "SEED" is used instead of the word "key", and sometimes other terms such as "Extended" Public/Private Key are used (N.T. Courtois, 2014; Courtois3b, 2015). In this paper we prefer to use the term of "Master" keys.

Extensions. It possible to see that such solutions can be deployed at several levels and lead to Hierarchical Deterministic (HD) solutions where more complex multi-level diagrams will also commute, cf. (N.T. Courtois, 2014; Courtois3b, 2015; Wuille, 2014). Such systems can be studied in terms of "security domains" or partially ordered sets which allows for information flow analysis cf. (N.T. Courtois, 2014).

2 KEY MANAGEMENT WITH AUDIT SOLUTIONS

The main idea in constructing such schemes [attributed to Greg Maxwell and improved/developed by Peter Wuille] is as follows, cf. (Wuille, 2014). Both derivation functions are essentially the same and have the same inputs or other "higher sensitive" inputs which allow to derive them. Then in the "Private Type 2 Derivation" function the private key is simply used to compute the corresponding public key and algebraic properties of ECC crypto are used. Following (N.T. Courtois, 2014) there are two **distinct** major ways to achieve this objective, a multiplicative one and an additive one.

Notation. In what follows we will assume that we work with an elliptic curve group generator G of $E(\mathbb{F}_P)$ with Q elements, where P, Q are two large primes. We will use small letters for private keys (integers $\mod Q$) and capital letters for public keys $\in E(\mathbb{F}_P)$. We will write $\hat{k}$ and $\hat{K}$ for master private/public keys. We will then write the application of "Private CKD" function $k(x)$ instead of SK_x in (N.T. Courtois, 2014) and Fig. 1. Then let $K(x)$ be the corresponding "Public CKD" function which would be denoted by PK_x in (N.T. Courtois, 2014). Our new notations emphasize the fact that x will no longer be a low-entropy index i but can be a quantity with a larger entropy.

Now we can propose the two basic methods and prove their correctness.

Theorem 2.0.1 (Two Basic Methods). *The* **multiplicative method** *based on (older[2]) Solution 1 of*

[2] Both solutions have been implemented and widely used in bitcoin community. Their history goes back to June 2011, a date when the forum thread "Deterministic Wallets" was started by Greg Maxwell at https://bitcointalk.org/index.php?topic=19137.0. The multiplicative solution was proposed earlier and was used in various systems such as Electrum. Since April 2013 there was a shift towards the additive method, claimed to be faster and easier to implement (Wuille, 2014) which became stan-

(N.T. Courtois, 2014) has Private CKD function:

$$k(x) = \hat{k} \cdot H(\hat{k}.G, x) \mod Q$$

The **additive method** *based on (more recent[2]) Solution 2 of (N.T. Courtois, 2014) is:*

$$k(x) = \hat{k} + H(\hat{k}.G, x) \mod Q$$

where H is some hash function with output reduced $\mod Q$. *Now in the* **multiplicative method** *we use the Public CKD:*

$$K(x) = H(\hat{K}, x).\hat{K} \ \ in \ \ E(\mathbb{F}_P)$$

where $\hat{K} = \hat{k}.G$ and for the **additive method** *we use:*

$$K(x) = \hat{K} + H(\hat{K}, x).G \ \ in \ \ E(\mathbb{F}_P)$$

Then *for each method both CKD functions match, i.e. $K(x) = k(x).G$.*

Proof: We write the proof to see what is that makes the result true. In the multiplicative solution we have

$$k(x).G = (\hat{k} \cdot H(\hat{k}.G, x)).G =$$

$$(H(\hat{k}.G, x)).(\hat{k}.G) = (H(\hat{K}, x)).\hat{K} = K(x).$$

the property we needed is that $a.(b.P) = b.(a.P)$ which we could call the scalar property or DH property. For the additive solution we have

$$k(x).G = (\hat{k} + H(\hat{k}.G, x)).G =$$

$$\hat{k}.G + H(\hat{k}.G, x).G = \hat{K} + H(\hat{K}, x).G = K(x).$$

the property we needed was that $(a+b).P = a.P + b.P$ a.k.a. distributive (or homomorphic) property. We see that both methods work and use two distinct algebraic properties of elliptic curve cryptography in order to obtain the same sort of result.

2.0.1 New Combined Method

It appears that nobody have yet noticed that both methods can be combined and that we can have a combined key derivation technique with a combined correctness result as follows:

Theorem 2.0.2 (Combined Method). *We assume that $\hat{K} = \hat{k}.G$ for the master secrets. We define a derived private key by:*

$$k(x) = \hat{k} \cdot H(\hat{k}.G, x) + H'(\hat{k}.G, x) \mod Q$$

where H, H' are two different hash functions with output reduced $\mod Q$. *The corresponding public key will be then:*

$$K(x) = H(\hat{K}, x).\hat{K} + H'(\hat{K}, x).G \ \ in \ \ E(\mathbb{F}_P)$$

For this new combined method we also have $K(x) = k(x).G$.

Proof: Proof needs simply to combine the two proofs above, one with H and other with H' to prove the equality of both parts independently.

dardized inside BIP032 specification. Later in 2014-15 a multi-key multiplicative method is shown to bring extra robustness (G. Gutoski, 2015).

2.1 Simple Exploit and More Robust Methods

In all these schemes we have the following well known [folklore] privilege escalation attack (N.T. Courtois, 2014; Courtois3b, 2015; G. Gutoski, 2015) in which one single derived key $k(x)$ and the higher level public key $\hat{K}$ allows to recompute the master private key $\hat{k}$. If we show it for our new combined scheme it will work also for earlier schemes which can be seen as special cases. Here is the formula which allows the attacker to recover the master private key: $\hat{k} = \frac{H'(\hat{k}.G,x) - k(x)}{H(\hat{k}.G,x)} \mod Q$. In 2014-15 Gutoski and Stebila proposed a multi-key multiplicative key management technique which allows to avoid this attack (G. Gutoski, 2015). A similar technique designed for our (different) purpose and against a wider range of attacks will be proposed in Section 6 below.

3 STEALTH ADDRESS KEY MANAGEMENT METHODS

The most basic Stealth Address (SA) technique was invented by user 'bytecoin' in bitcoin forum on 17 April 2011 (user 'bytecoin', 2011). Improved variants were proposed later in 2013-14 (van Saberhagen, 2013; Todd, 2014) which we study below in Section 3.1.

The goal of all Stealth Address (SA) methods is to send money to a certain "publicly visible" master key in such a way that this key does **not** appear in the blockchain. For this purpose, other seemingly totally unrelated keys are used, and in their essence all these methods are key management techniques with additional secrets or/and randomness very similar to those we have studied so far. We first summarize and examine the exact original method of (user 'bytecoin', 2011) using the notations of slide 21 in (Courtois6, 2016).

1. The recipient has a public key $B = b.G$
2. The sender uses public key $A = a.G$
3. Now Diffie-Hellman allows both the sender and the recipient to compute a certain value S.

$$S = a.B = b.A \ \ \in E(\mathbb{F}_P)$$

4. The ephemeral transfer address is then simply $H(S).G$ in $E(\mathbb{F}_P)$, private key is $c = H(S) \mod Q$ and in normal bitcoin operation only $H'(H(S).G)$ would be revealed initially when coins are sent to $pk = H(S).G$.

5. The receiver actively monitors the blockchain or other channels for all plausible A and checks if somebody is sending coins to some $H'(H(b.A).G)$. He can spend all such outputs.

This original 2011 method contains two serious mistakes which affect both security and privacy. The **first mistake** is to use words 'by his private key' for the sender which wrongly suggests or implies the usage of a permanent identity A, or it is not clear on this and allows the developers to get it wrong. Now if the sender uses this A more than once or if this key A is in any way related to his other actions in the network, this is not a good idea, cf. slides 15-38 in (Courtois6, 2016). Later specifications such as (van Saberhagen, 2013) and Todd post (Todd, 2014) make it clear we need to use a random ephemeral 'nonce keypair' here denoted[3] as r,R. In this case the value R must be somewhat transmitted with the transaction which increases the blockchain space required. A popular modern method to publish extra data in bitcoin is to use the OP_RETURN instruction which allows to put arbitrary data in outputs of bitcoin transactions.

A second mistake in the original method which was also already discussed in 2011 (user 'bytecoin', 2011) and fixed in all later proposals (van Saberhagen, 2013; Todd, 2014), is that in this scheme both the sender and the receiver can spend. Both can compute the ephemeral private key c, cf. (user 'bytecoin', 2011; Todd, 2014; Courtois6, 2016). Therefore if the receiver does not spend them immediately or is offline, the sender can change his mind and take his money back.

3.1 Improved Basic Stealth Address Method

Fixing these 2 problems leads to an improved basic method which is basically extended by an additive key management technique in the sense of Thm. 2.0.1. where a DH public key mechanism is used to derive the private key offset. This method is frequently claimed to be designed in Jan. 2014 by Todd in (Todd, 2014) and is also described in (dev. team Darkwallet, 2014) and on slides 27-29 in (Courtois6, 2016) if we rename a,A by r,R. In fact it was clearly known earlier and the same exact method with exactly the same formulas was already earlier described by Nicolas van Saberhagen in CryptoNote white paper in Oct. 2013 (van Saberhagen, 2013). Only later on 6 January 2014 it was adapted[4] by Todd to make it work within bitcoin spec (Todd, 2014).

1. The recipient has a public key $B = b.G$
2. The sender uses a one-time nonce pair $R = r.G$, $r \leftarrow$ random $\mod Q$.
3. Diffie-Hellman allows both to compute the same value $c = H(S)$:

$$c = H(S) = H(r.B) = H(b.R) \mod Q$$

4. The ephemeral private key which only the receiver can compute is then:

$$c + b = H(b.R) + b \mod Q$$

and the publicly visible address which will appear on the blockchain (which all three: sender, auditor and receiver can compute) is the address with public key equal to $B + H(S).G$ and:

$$H(S).G + B = H(r.B).G + B = H(b.R).G + b.G$$

5. The receiver actively monitors the blockchain for transactions which included a publication of some R value for example after an OP_RETURN, and for such transactions he can compute the private key and spend.

Speed vs. Privacy Variants. In (Todd, 2014) Todd suggests several additional methods to publish a few bits of extra information together with R in order to do blockchain scanning faster.

3.2 The Question of Audit and View-only Wallets

Until now the SA solutions have a serious problem: the same entity has to know the private 'spend' key b and scan the blockchain in the real-time for some pairs $H(S).G + B, R$. This is contrary to very common practice of cold/disconnected storage of private keys. One very simple solution to this problem is called "Non-P2SH-Multisig stealth" in (dev. team Darkwallet, 2014). The solution proposed is to use the previous method twice, with $B = b.G$ and $B' = b'.G$ and B' is advertised together with B as a (twice longer) stealth address. Here b' can be held in cold storage and will not be needed to check for incoming payments. This is achieved by using multi-sig on the top of $PK1 = H(S).G + B$ transfer key which is not used

[3] The notation used is $P = e.G$ in (Todd, 2014; dev. team Darkwallet, 2014) and $R = f.G$ in DarkWallet (dev. team Darkwallet, 2014) and $P = e.G$ in ShadowCash source code at github.com/shadowproject. Our preferred notation is rather $R = r.G$ as in (van Saberhagen, 2013; Courtois6, 2016).

[4] Todd clearly says that the orignal (more basic and partly flawed) idea comes from 2011 'bytecoin' post and also credits Maxwell, Back, Taaki and others for inputs, yet he omits to mention (van Saberhagen, 2013).

directly. Instead the sender sends coins to a 2-out-of-2 multi-sig address of type $PK1, PK2$, i.e. two keys are needed to spend. The nonce r which is called e in (dev. team Darkwallet, 2014) is proposed to be the same in both cases. We have:

$$PK1 = H(r.B).G + B = H(b.R).G + b.G \quad \in E(\mathbb{F}_P)$$

$$PK2 = H(r.B').G + B' = H(b'.R).G + b'.G \quad \in E(\mathbb{F}_P)$$

This can be extended to more general multisig scenarios. To the best of our knowledge, using the same nonce r twice here is NOT a problem in this method of (dev. team Darkwallet, 2014). Re-using r's is also encouraged on p. 7 of (van Saberhagen, 2013).

Disambiguation. The method above, although it uses two keys is NOT what is commonly called Dual-key or 2-key SA, which do not require any multi-sig, and which we study below.

4 DUAL-KEY IMPROVED STEALTH ADDRESS METHODS

An important enhancement to SA methods is due to a developer known as rynomster/sdcoin who has on 2/08/2014 announced a first full working implementation of Dual-key SA in ShadowCash at https://bitcointalk.org/ index.php? topic=700087.msg 8153845. Here is a short description of Dual-key SA method which is used in many current systems [Monero,DarkWallet,ShadowCash,etc] and is described in Section 'Dual-key stealth' of (dev. team Darkwallet, 2014) and on slides 31-40 in (Courtois6, 2016).

1. The recipient has Stealth Address SA in the form of two public keys hence Dual-key name. We have a 'scan' public key V and a 'spend' public key B using vocabulary of (sx library, 2013; dev. team Darkwallet, 2014). We have $V = v.G$ and $B = b.G$ using the notations on slides 31-40 in (Courtois6, 2016). ECC points V, B have 33 bytes typically, cf. (sx library, 2013), the scalars v, b require only 32 bytes.
2. We have $V = v.G$ and we call V a 'scan pubkey' (sx library, 2013) or 'View key' cf. (Courtois6, 2016). We have $B = b.G$ and we call B a 'spend pubkey' (sx library, 2013; dev. team Darkwallet, 2014).
3. The key advertised by the receiver of coins is B, V. None of these keys ever appears in the blockchain, only the sender and the receiver know B, V.
4. The sender uses a one-time nonce pair $R = r.G$, $r \leftarrow$ random $\mod Q$.
5. Diffie-Hellman allows both the sender and the recipient to compute the same value $c = H(S)$:
$$c = H(S) = H(r.V) = H(v.R) \quad \mod Q$$
6. The ephemeral private key which only the receiver can compute is:
$$c + b = H(v.R) + b \quad \mod Q$$
and the publicly visible address which will appear on the blockchain (which all three: sender, auditor and receiver can compute) is again $B + H(S).G$:
$$H(S).G + B = H(r.V).G + B = H(v.R).G + b.G$$
7. The auditor, hot wallet, proxy server or read-only wallet knows/has the pair B, v.
8. The auditor actively monitors the blockchain for transactions which included a publication of some R value for example after an OP_RETURN, and for such transactions he can compute
$$pk = H(v.R).G + B \quad \in E(\mathbb{F}_P)$$
and see if this pk or its hash appears in the blockchain.
9. The auditor is not able to spend coins because he does not know b. Only the recipient knows b and can compute $sk = H(v.R) + b \mod Q$ and spend these bitcoins.

4.0.1 Dual-key SA In Practice

Details on how this can be done in practice in bitcoin can be found in (dev. team Darkwallet, 2014; sx library, 2013). For Monero an interactive tool is available at https://xmr.llcoins.net/addresstests.html. In ShadowCash the functionality is implemented inside StealthSecretSpend() specified in https://github.com/shadowproject/shadow/blob/master/src/stealth.cpp.

4.1 Dual-key SA Security and Privacy

It is worth noting that the Dual-key SA is provides very strong anonymity for receivers and unlinkability of different payments received by the same receiver. Thus "users can receive CryptoNote-based cryptocurrencies with no concern for their privacy" (A. Mackenzie, 2015) for Monero, and the same applies to ShadowCash and to bitcoin users who use DarkWallet (dev. team Darkwallet, 2014). Unhappily this property is frequently violated by users themselves or by their software wallets see (A. Mackenzie, 2015), this as soon as they spend the moneys received. When various attributions are

later inputs to the same transaction there is a good chance that amounts sent to different PK belonging to the same user will be later merged and thus linked together in the blockchain (DarkWallet provides additional mixing (dev. team Darkwallet, 2014)).

CryptoNote De-Anonymising Attacks. An interesting question in all SA schemes is whether a bugged or malicious choice of the hash function H could make these schemes less secure. This has recently happened in ShadowCash extension of CryptoNote protocol where our Dual-key SA is combined with a ring signature scheme cf. https://archive.is/3VEHr.

5 A BAD RANDOM ATTACK

In this section we describe a simple attack which shows that a solution more robust than those described above may be required. It is going to be a combination attack similar as in (N.T. Courtois, 2014). We refer to (N.T. Courtois, 2014) and blog.bettercrypto.com/?page_id=1467 for statistics about bad random events in bitcoin. The key point is that bad random attacks are not always very strong. For example if the same random r is used twice in two different signatures the only attack in such case is the each of the two users can recover the other user's public key, cf. Proposition 43 in (N.T. Courtois, 2014). Now additional attacks in which anyone can recover their keys are possible in some configurations, see for example Proposition 45 in (N.T. Courtois, 2014). Furthermore many more additional attacks are possible if user keys are derived using key popular key management techniques such as those we studied here in Section 2 or HD Wallet/BIP032 techniques (Wuille, 2014) Then we have a larger of possible attacks, cf. Sections 6-10 in (N.T. Courtois, 2014). In this paper we show that similar attacks are also possible with Stealth Address methods.

Theorem 5.0.1 (Combination Attack On Dual-key Wallets). *If the attacker knows the audit key B, v for one recipient AND if two identical randoms are used just once in any pair of transactions sent to this recipients, then we can recover the private key b which allows to spend all coins ever sent to B, V.*

Proof: We recall how a standard ECDSA signature works. We pick a random non-zero number $a \mod Q$ and the signature of m is the pair u, s with

$$u = (a.G)_x; \;\; s = (H(m) + ku)/a \mod Q$$

Now in our attack we have:

$$\begin{aligned} as &= (H(m) + uk) \mod Q \\ as' &= (H(m') + uk') \mod Q \\ k &= H(v.R) + b \mod Q \\ k' &= H(v.R') + b \mod Q \end{aligned}$$

So we have:

$$u(H(v.R) - H(v.R')) = a(s - s') + (H(m) - H(m'))$$

This equation allows to compute u by division mod Q and then the first two equations allow to compute k and k' which given the last two equations give b and b'.

6 A NEW ROBUST STEALTH ADDRESS TECHNIQUE

We are now going to define a new particularly robust stealth payment technique which combines the ideas of using both the additive and multiplicative technique of Thm. 2.0.2 and the idea of a multi-key multiplicative technique of (G. Gutoski, 2015) in order to be resistant to key leakage and other attacks such as above.

1. The recipient will have $m+1$ private/public key pairs. We have one 'scan pubkey' or a.k.a. 'View key' $V = v.G.$ and we have m different 'spend' public keys $B_i.$ and $B_i = b_i.G.$
2. The key advertised by the receiver of coins is $B_1 \ldots B_m, V.$ None of these need to appear in the blockchain and only the sender and the receiver need to know them.
3. The sender uses a one-time nonce pair $R = r.G.$
4. Diffie-Hellman allows both the sender and the recipient to compute the same value S:
$$S = r.V = v.R \;\; \in E(\mathbb{F}_P)$$
5. We assume that we have an expanding hash function $H_0, \ldots H_m$ with $m+1$ outputs which are all numbers mod Q, which can be implemented as a combination of a standard hash function and a stream cipher or RNG.
6. The ephemeral private key which only the receiver can compute is:
$$H_0(v.R) + \sum H_i(S).b_i \mod Q$$
and the publicly visible address which will appear on the blockchain (which all three: sender, auditor and receiver can compute) is now:
$$H_0(S).G + \sum H_i(S).B_i \;\; \in E(\mathbb{F}_P)$$

7. The auditor, hot wallet, proxy server or read-only wallet contains/knows $m+1$ values B_i and the secret key v. Again he actively monitors the blockchain for transactions which included some R value and for such transactions he can compute

$$pk = H(v.R).G + \sum H_i(v.R).B_i \quad \in E(\mathbb{F}_P)$$

and see if this pk or its hash appears in the blockchain. Auditor is not able to spend coins because he does not know the b_i.

Now we are going to specify our security assumption (same as in (G. Gutoski, 2015)):

Definition 6.1 (EC 1MDLP Problem). *We consider all the attackers as follows. The attacker is a probabilistic Turing machine M with bounded computations with access to two oracles. The first is a challenge oracle which produces m random elements $Q_i \in E(\mathbb{F}_P)$. The second oracle allows to solves the ECDL problem for up to $m-1$ queries chosen by the attacker machine M. We say that M wins if it is able to output the discrete logarithms for all m elements Q_i provided by the first oracle.*

Now we claim that:

Theorem 6.0.1 (Security of Robust SA Method). *Our new robust stealth payment scheme allows to protect anonymity of users and protect the spending keys against thefts even when the attacker can recover*[5] *up to m_1 individual spending private keys and if up to m_2 bad*[6] *randoms were used in ECDSA spending transaction with any $m_1 + m_2 < m$. If an attacker can break our payment scheme, one can (efficiently) convert it into an oracle solving EC 1MDLP. (G. Gutoski, 2015).*

Proof [sketch]: With a recovery of up to m_1 individual spending private keys and up to m_2 repeated/bad/related randoms we can hope to obtain at most $m_1 + m_2 < m$ linear equations which involve at least m variables b_i by formulas such as in the proof of Thm. 5.0.1. This remains insufficient to solve a linear system of equations and leads to a situation identical as in the proof of main Thm. in (G. Gutoski, 2015). If in our robust stealth payment scheme, all the m private spend keys can be recovered by a certain attacker M we can argue by game hopping that the attacker should also be able to recover m private keys with the knowledge of $m-1$ discrete logs from an oracle, querying these specific combinations. This is believed to be a hard problem cf. (G. Gutoski, 2015).

[5] Could be due to malware, side channel attacks, brain wallets (N.T. Courtois, 2016) or other from of leakage or compromise.

[6] Random numbers can be repeated, guessed due low entropy or related to each other, cf. Section 5 in (N.T. Courtois, 2014).

7 CONCLUSION

In this paper we review the principal key management and Stealth Address techniques which have been invented in the recent years and are used in numerous crypto currency and blockchain wallets and systems. We show their correctness, discuss additional variants, and show that some techniques offer yet a limited level of privacy and security. In addition we show that one can do better than the Dual-key Stealth Address technique which is the one which is used in many current systems such as Monero or DarkWallet. We propose a new improved SA technique which was designed to be more robust against a variety of attacks. Our new method is resistant to the leakage/compromise of one or several private keys. It can also resist to other incidents at operation such as bad-random events. The price to pay for this is an m-fold increase in the size of the higher level public keys. The size of the actual transactions which need to be published on the blockchain does not need to increase.

REFERENCES

A. Mackenzie, S. Noether, M. C. T. (2015). Improving obfuscation in the cryptonote protocol. In *online paper*. https://lab.getmonero.org/pubs/MRL-0004.pdf.

Courtois3b, N. (2015). Bitcoin key management: Hd wallets, bip032. In *slides*. http://www.nicolascourtois.com/bitcoin/paycoin_dig_sign_key_mng_HD_BIP032_3b.pdf.

Courtois3d, N. (2015). What bitcoin private keys say to each other. In *slides*. http://www.nicolascourtois.com/bitcoin/paycoin_dig_sign_combination_attacks_cold_storage_3d.pdf.

Courtois6, N. (2016). Anonymous crypto currency, stealth address, ring signatures, monero, comparison to zero.cash. In *slides*. http://www.nicolascourtois.com/bitcoin/paycoin_privacy_monero_6.pdf.

dev. team Darkwallet (2014). Darkwallet/stealth. In *part of Dark Wallet public development wiki*. https://wiki.unsystem.net/en/index.php/DarkWallet/Stealth.

G. Gutoski, D. S. (2015). Hierarchical deterministic bitcoin wallets that tolerate key leakage. In *Financial Cryptography*, volume LNCS 8975, pages 497–504. https://eprint.iacr.org/2014/998.

N.T. Courtois, P. Emirdag, F. V. (2014). Private key recovery combination attacks: On extreme fragility of popular bitcoin key management, wallet and cold storage solutions in presence of poor rng events. In *Eprint*. http://eprint.iacr.org/2014/848.

N.T. Courtois, G. Song, R. C. (2016). Speed optimizations in bitcoin key recovery attacks. In *will appear in proc. of CECC 2016*. https://eprint.iacr.org/2016/103.pdf.

S. Eskandari, D. Barrera, E. S. J. C. (2015). A first look at the usability of bitcoin key management. In *In Workshop on Usable Security (USEC)*. https://people.inf.ethz.ch/barrerad/files/usec15-eskandari.pdf.

sx library (2013). Stealth payments section 9,. In *part of online manual for "sx library"*. https://sx.dyne.org/stealth.html.

Todd, P. (2014). [bitcoin-development] stealth addresses. In *post of 04:06:05 -0800 Mon 06 Jan*. http://www.mail-archive.com/bitcoin-development@lists.sourceforge.net/msg03613.html.

user 'bytecoin', A. (17 April 2011). Untraceable transactions which can contain a secure message are inevitable. https://bitcointalk.org/index.php?topic=5965.0.

van Saberhagen, N. (2013). Cryptonote v 2.0. In *online paper*. https://cryptonote.org/whitepaper.pdf.

Wuille, P. (2014). Bip032 description, 15 jan. In *the official specification of BIP032*. https://github.com/bitcoin/bips/blob/master/bip-0032.mediawiki.