Uncontrolled Randomness in Blockchains: Covert Bulletin Board for Illicit Activity

Nasser Alsalami
Lancaster University, UK
n.alsalami@lancaster.ac.uk

Bingsheng Zhang
Zhejiang University, China
bingsheng@zju.edu.cn

***Abstract*—Public blockchains can be abused to covertly store and disseminate potentially harmful digital content which poses a serious regulatory issue. In this work, we show the severity of the problem by demonstrating that blockchains can be exploited to surreptitiously distribute arbitrary content. More specifically, all major blockchain systems use randomized cryptographic primitives, such as digital signatures and non-interactive zero-knowledge proofs; we illustrate how the uncontrolled randomness in such primitives can be maliciously manipulated to enable covert communication and hidden persistent storage. To clarify the potential risk, we design, implement and evaluate our technique against the widely-used ECDSA signature scheme, the CryptoNote's ring signature scheme, and Monero's ring confidential transactions. Importantly, the significance of the demonstrated attacks stems from their undetectability, their adverse effect on the future of decentralized blockchains, and their serious repercussions on users' privacy and crypto funds. Finally, we present a generic framework to immunize blockchains against these attacks.**

I. Introduction

The blockchain technology has pioneered a new paradigm to realize large-scale distributed ledgers. While the blockchain technology is promising in a great number of application scenarios, it can also be abused to anonymously store and disseminate potentially harmful digital content. As shown in a recent study [1], 1.4% Bitcoin transactions contain non-financial data, some of which contain objectionable content. Though the absence of a central censor makes blockchains appealing in some use cases, the increasing amount of illicit content posted to the blockchains poses a serious regulatory issue [2]. Subsequently, several techniques have been discussed to either filter unwanted content before posting to the ledger [3] or remove content from the blockchain [4] [5].

However, all of the proposed countermeasures are only effective if the malicious content attached to the transactions can be detected. The situation gets worse when the attackers hide data into normal transactions and use blockchain platforms for covert communications. Naively, one can encrypt the malicious content and attach its ciphertext to a transaction, but it is noticeable to the public. In 2018, Partala [6] showed a proof-of-concept steganography technique that allows an adversary to covertly embed one bit into a standard Bitcoin transaction's recipient address without being distinguished from an innocuous transaction and without burning the funds.

In this work, we further advance this line of research by demonstrating an effective steganographic method that offers high throughput and can be launched against any blockchain platforms that use randomized cryptographic primitives, such as digital signatures and non-interactive zero-knowledge proofs. The main observation is that all randomized algorithms need to consume random coins somewhere along the execution, and these random coins are not audited or certified publicly. By intentionally manipulating the random coin supplied to a randomized algorithm, an attacker is able to embed arbitrary information into the output of the algorithm, where the output that contains steganographic data is computationally indistinguishable from normal output.

Our attack can be used for covert channels, persistent storage, and many other scenarios. For instance, the attacker(s) may try to subvert, or mis-implement cryptocurrency wallets and re-distribute them to unsuspecting users. The subverted wallets can then surreptitiously leak the victim's secret, such as the signing key, via standard transactions. Importantly, the transactions generated by the subverted wallets are computationally indistinguishable from normal transactions for any third-party observers. Note that the current focus of research regarding blockchain subversion vulnerabilities is mainly on the trusted parameter setup process, such as common reference string (CRS) generation [7] [8], while software/hardware subversion vulnerabilities in blockchain cryptocurrency applications has not been extensively studied.

We would like to argue that wallet subversion is practical and plausible, and the public should be alert to such risks. Cryptocurrencies have very complex cryptographic primitives and structures that make them prone to unseen mis-implementations. Although many cryptocurrencies are marketed as decentralized projects, studies have found that the development of many blockchain applications is highly centralized. For example, 30% of the source files in Bitcoin are written by a single author, and 7% of the code is written by the same author [9]. Similarly, 20% of the source code in Ethereum is attributed to the same author [9]. This high centralization may cause bias and introduce intentional/unintentional flaws. Moreover, most end users lack the ability and the means to check the conformity of an executable wallet with its reference source code. It is uncommon for users to compile the source code of any application by themselves; instead, they usually relay on downloading readily prepared executable applications. The difficulty to examine the implementation of a cryptocurrency wallet is even more pertinent to hardware wallets, such as the various *Swiss-Army-Knife* hardware wallets [10]. These hardware wallets are typically manufactured in an outsourced loosely-controlled environment, and it is virtually impossible to audit the integrity of their implementation through the standard functionality 'correctness' test by observing input/output pairs in a black-

box manner.

Our contributions. The primary objective of this work is to draw attention to the potential threat of abusing uncontrolled randomness in blockchain cryptographic algorithms. To the best of our knowledge, this is the first work in literature that discusses such a widely spread vulnerability in the blockchain context. We summarize our contributions as follows:

A novel blockchain steganographic technique. We propose a new steganographic technique that affects most cryptocurrencies. Our technique greatly increases the throughput of the state-of-the-art blockchain steganographic attacks. We present our general attack against the widely-used CryptoNote framework, and we implement and evaluate the attack on two real world cryptocurrencies – Monero and Bytecoin.

A covert broadcast communication system. We designed and implemented the world's first practical covert broadcast communication system by integrating our steganographic technique with Boneh et al. broadcast encryption scheme [11]. The prototype system is deployed on Bytecoin, and it can be used to bypass any censorship mechanisms, as the mere existence of such a channel is unobservable.

Persistent storage. With the proposed steganographic technique, anyone can use the blockchain as a cheap hidden persistent storage along with their daily transactions. For instance, this can be used for uncensorable cyberlockers. At the time of submission, persistently storing 1GB of data on Bytecoin blockchain costs less than $3.

Wallet subversion attacks. For the first time, we point out that there is a troubling risk of massive coin stealing among all of the current cryptocurrency wallets by demonstrating efficient and effective subversion attacks within the realm of *Kleptography* and *Algorithm Substitution* Attacks. This attack possesses the following properties:

- *Passive attack.* After the victim user downloads and installs the subverted wallet, the attacker does not need to interact directly with the victim's wallet. The communication channel between the subverted wallets and the attacker is through the posted blockchain transactions.
- *Undetectability.* The transactions generated by the subverted wallets are computationally indistinguishable from the honestly-generated ones. Therefore, no watchdog can detect the subversion in the black-box setting.
- *Interoperability.* The subverted wallets transact seamlessly with normal wallets, i.e. they can send to and receive from other wallets regardless whether other wallets are subverted.

We have implemented our subversion attacks against the ECDSA signature scheme, and the ring signature used in the CryptoNote framework which is implemented by many cryptocurrencies, such as Bytecoin and Monero. As a result, this work has direct impact on 18 of the top 25 cryptocurrencies in terms of market capitalization [12] as depicted in Table I.

Countermeasures. We propose a stego-resistant blockchain framework to prevent our blockchain steganographic attacks and mitigate the wallet subversion attacks. Intuitively, in the proposed framework, the miners are trusted, and upon receiving (randomized) signature associated with a transaction, instead of directly including it into the transactions in the next block, the miner either re-randomizes the signature or replaces the signature with an non-interactive zero-knowledge proof showing that the miner has seen a valid signature for the transaction. Subsequently, the possibly stego-generated signatures are dropped.

Cryptocurrencies' Signatures					
#	**Cryptocurrency**	**ECDSA**	**EdDSA**	**Ring Signature**	**Note**
1	Bitcoin	✓			
2	Ethereum	✓			
3	Ripple	✓	✓		
4	Bitcoin Cash	✓			
5	Litecoin	✓			
6	Cardanos		✓		
7	Stellar		✓		
8	Zcash		✓		
9	IOTA				Winternitz
10	Monero			✓	
11	Dash	✓			
12	NEM		✓		
13	Ethereum Classic	✓			
14	Komodo	✓			
15	Verge	✓			
16	Lisk		✓		
17	Dogecoin	✓			
18	Decred	✓	✓		
19	Nano		✓		
20	Wanchain	✓		✓	
21	Bytecoin			✓	
22	Siacoin		✓		
23	Bitcoin Diamond	✓			
24	BitShares	✓			
25	Waves		✓		

TABLE I: Top 25 cryptocurrencies (currencies checked with either ECDSA or ring signature are effected by our attacks.)

Discovered Bytecoin implementation bug. While examining the Bytecoin's wallet, we have observed a major discrepancy between the wallet execution behaviour and the CryptoNote specification. In particular, CryptoNote specifies that the signer's public key should be randomly positioned in the ring signature, whereas in implementation, the distribution of the signer's public key position is far from being uniformly random. As part of our responsible disclosure, we reported the bug to the Bytecoin development team who acknowledged the bug with bounty; however, to date, the bug still exists in the latest version (v 3.3.3). This implementation flaw is further explained in Sec. VIII.

II. Preliminaries

Notations. Unless specified in the context, we use $\mathsf{hash}_p : \{0,1\}^* \mapsto \mathbb{Z}_p$ and $\mathsf{hash}_g : \{0,1\}^* \mapsto \mathbb{G}$ as two collision resistant hash functions that map an arbitrary length string to a group element in $\mathbb{Z}_p$ and $\mathbb{G}$, respectively.

Brief description of CryptoNote. CryptoNote is a protocol proposed by Nicolas van Saberhagen [13], and it has been implemented in many emerging cryptocurrencies, such as Bytecoin [14], CryptoNoteCoin [15], Fantomcoin [16], etc. Compared to Bitcoin-like cryptocurrencies, CryptoNote offers two main features: (i) *stealth address via non-interactive key exchange* and (ii) *set anonymity via (linkable) ring signatures*.

Precisely, the user's private key consists of $a, b \in \mathbb{Z}_p$, and the corresponding public key consists of $A := g^a$ and $B := g^b$. Note that in a standard CryptoNote implementation,

CryptoNote Signing Algorithm

$\mathsf{Sign}(\{P_j\}_{j=1}^k, t_\ell, \ell, m)$:

- Set $I := \mathsf{hash}_g(P_\ell)$;
- For $j \in [k]$, pick $q_j \xleftarrow{\$} \mathbb{Z}_p$;
- For $j \in [k], j \neq \ell$, pick $w_j \xleftarrow{\$} \mathbb{Z}_p$;
- For $j \in [k]$:
  - Set $L_j := g^{q_j}$ if $j = \ell$;
    Set $L_j := g^{q_j} \cdot P_j^{w_j}$ if $j \neq \ell$;
  - Set $R_j := (\mathsf{hash}_g(P_j))^{q_j}$ if $j = \ell$;
    Set $R_j := (\mathsf{hash}_g(P_j))^{q_j} \cdot I^{w_j}$ if $j \neq \ell$;
- Set $c := \mathsf{hash}_p(m, L_1, \ldots, L_k, R_1, \ldots, R_k)$;
- For $j \in [k]$:
  - Set $c_j := w_j$ if $j \neq \ell$;
    Set $c_j := c - \sum_{j=1}^k c_j$ if $j = \ell$;
  - Set $r_j := q_j$ if $j \neq \ell$;
    Set $r_j := q_\ell - c_\ell t_\ell$ if $j = \ell$;
- Return $\sigma := (I, c_1, \ldots, c_k, r_1, \ldots, r_k)$.

Fig. 1: CryptoNote Signing Algorithm.

a is usually defined as $\mathsf{hash}_p(b)$; therefore, b is the actual secret key. To transfer funds to a recipient, the payer needs to generate a transaction public key $R := g^r$ and compute the corresponding one-time address $T := g^{\mathsf{hash}_p(A^r)} \cdot B$. The recipient is then able to compute the corresponding one-time private key as $t := \mathsf{hash}_p(R^a) + b$. Notice that $A^r = R^a$. With regards to the ring signature schemes, it is transformed from the OR-composition of Schnorr's identification Sigma protocols. There exists a LNK algorithm that can link two signatures together if they are produced by the same signing key. By design, the one-time signature key can only be used once, and it can be detected if the same key is used to sign two transactions, which prevents double spending. In particular, let $T := g^t$ be the one-time public key, and define $I := (\mathsf{hash}_g(T))^t$ as a "key image" as part of the signature. The ring signatures signed by the same secret key would have identical key image; therefore, double spending can be defeated efficiently by simply checking if the key image has already been used. As depicted in Fig. 1, the signing algorithm takes input as a set of public keys $\{P_j\}_{j=1}^k$ (a.k.a. the ring), the secret key t_ℓ such that $P_\ell = g^{t_\ell}$, $\ell \in [k]$.

Brief description of Monero (Version 0.12.0.0). Monero [17] is one of the most successful CryptoNote-based cryptocurrencies. Although the original Monero was based on the CryptoNote protocol, its transaction signature has evolved beyond this protocol[1]. As mentioned in [19], CryptoNote suffers from a shortcoming where amounts in transactions are not hidden. To address this issue, *Ring Confidential Transaction* (RingCT) [19] has been developed and deployed in Monero since January 2017. It combines (linkable) ring signature and Pedersen commitment schemes [20], and also adopts Multi-layered Linkable Spontaneous Anonymous Group Signature.

[1] The Monero project is very active and evolves rapidly. In fact they have two major releases each year. In Oct. 2018, Monero released version 0.13.0.0 "Beryllium Bullet", which switched to Bulletproofs [18]. Since the technical specification of the latest version is not well documented yet, our work is for Monero version 0.12.0.0 and earlier versions.

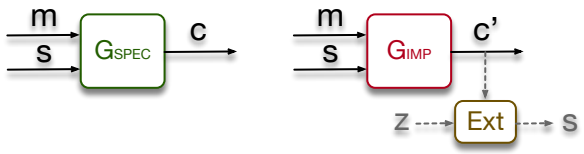


Fig. 2: Kleptography/ASA: specification $\mathsf{G}_{\mathsf{SPEC}}$ takes input as the message m and the secret s, and outputs c; whereas, the malicious implementation $\mathsf{G}_{\mathsf{IMP}}$ outputs a subverted ciphertext c' which can leak the secret s exclusively to the attacker who knows z.

Steganography. Steganography refers to the techniques that allow a sender to send a message covertly over a *communication channel* so that the mere presence of the hidden message is not detectable by an adversary who monitors the channel [21] [22]. Let $\mathcal{C}$ be a channel on the alphabet Σ with length ℓ, which can be viewed as a function that maps the channel history $\mathcal{H} \in (\Sigma^{\leq \ell})^*$ to a probability distribution upon $\Sigma^{\leq \ell}$. Denote the probability distribution by $\mathcal{C}_{\mathcal{H}}$. A stegosystem on a family of channels $\mathbf{C} := \{\mathcal{C}^\lambda\}_{\lambda \in \mathbb{N}}$ consists of a triple of PPT algorithms $\mathcal{ST} := (\mathsf{KeyGen}, \mathsf{Encode}, \mathsf{Decode})$ as follows:

- $(\mathsf{ek}, \mathsf{dk}) \leftarrow \mathsf{KeyGen}(1^\lambda)$ is the key generation algorithm that takes as input the security parameter 1^λ, and it outputs an embedding key ek and an extraction key dk.
- $\mathsf{st} \leftarrow \mathsf{Encode}_{\mathsf{ek}}^{\mathcal{C}_{\mathcal{H}}^\lambda}(m, \mathcal{H})$. Given an embedding key ek, a message m, and a channel history $\mathcal{H}$. Encode generates a stegotext message st. Note that Encode has sampling access to $\mathcal{C}_{\mathcal{H}}^\lambda$.
- $m \leftarrow \mathsf{Decode}_{\mathsf{dk}}(\mathsf{st})$, Decode takes as input an extraction key dk and the stegotext st, and outputs the hidden message $m \in \{0,1\}^*$.

Definition 1. *We say a stegosystem $\mathcal{ST}$ is* indistinguishable chosen-hiddentext-attack *(IND-CHA) secure if for all PPT adversary $\mathcal{A}$ we have*

$$\left| \Pr \left[\begin{array}{l} (\mathsf{ek}, \mathsf{dk}) \leftarrow \mathsf{KeyGen}(1^\lambda); b \leftarrow \{0,1\}; \\ b^* \leftarrow \mathcal{A}^{\mathcal{O}(b,\cdot,\cdot)}(1^\lambda) : b = b^* \end{array} \right] - \frac{1}{2} \right| = \mathsf{negl}(\lambda)$$

where $\mathcal{O}(b, m, \mathcal{H})$ is the oracle that returns $\mathsf{st} \leftarrow \mathcal{C}_h^\lambda$ if $b = 0$; otherwise, it returns $\mathsf{st} \leftarrow \mathsf{Encode}_{\mathsf{ek}}^{\mathcal{C}_{\mathcal{H}}^\lambda}(m, \mathcal{H})$.

Kleptography/Algorithm-substitution attacks. Our wallet subversion attacks can be classified as kleptographic attacks [23]–[25] and algorithm-substitution attacks (ASA) [26] [27]. As a high-level definition, in such attacks, the adversary maliciously tampers with the implementation of a cryptographic algorithm $\mathsf{G}_{\mathsf{IMP}}$ and changes it from its specification $\mathsf{G}_{\mathsf{SPEC}}$ algorithm, with the aim to subliminally and exclusively leak the user's secret information to the adversary while evading detection in the black-box setting. The depiction in Fig. 2 illustrates how an adversarial implementation $\mathsf{G}_{\mathsf{IMP}}$ of the algorithm $\mathsf{G}_{\mathsf{SPEC}}$ can allow the adversary, given their secret key z, to detect the subverted output c' and extract the user's secret s. Kleptographic attacks are significant due to their undetectability in the black-box setting and their severe consequences on the security of the users.

III. Generic Steganographic Attack

Many cryptocurrencies use ring signatures to enhance users' privacy. For example, the CryptoNote framework [13], which is adopted by around 20 cryptocurrencies, uses ring signatures. As a demonstration, we describe how the *uncontrolled randomness* (marked in grey in Fig. 1) in CryptoNote's ring signature signing algorithm can be maliciously exploited. Precisely, we show how the randomness within the ring signatures can be used to communicate covertly, store arbitrary information, and surreptitiously leak private keys. Note that the same principles are applicable to any other uncontrolled randomness in blockchain cryptographic primitives.

Our generic steganographic attack on CryptoNote. The CryptoNote protocol uses the ED25519 twisted Edwards curve, and the group order is a 253-bit prime p. The long term secret key of a user consists of two group elements $a, b \in \mathbb{Z}_p^*$, but $a := \mathsf{hash}_p(b)$ is commonly used in practical implementation. Therefore, the long term secret key of a CryptoNote account is effectively 253 bits.

As part of the (linkable) ring signature, a *one-out-of-many* non-interactive zero knowledge proof is included. In addition, for a ring of size k, the format of the ring signature is $\sigma = (I, c_1, \ldots, c_k, r_1, \ldots, r_k)$. Suppose the sender's public key is PK_ℓ, $\ell \in [k]$. For all $j \in [k]$ and $j \neq \ell$, the components c_j and r_j are uncontrolled random group elements in $\mathbb{Z}_p$ and can be used for covert communication. Hence, our attack is premised on steganographically embedding arbitrary information on the ring signature's random numbers (c_j, r_j). In our attack example, $\mathsf{ek} = \mathsf{dk}$, which is the common shared secret $z \in \{0,1\}^{128}$. The attack is explained as a three-step process carried out by two parties: a sender called Alice and a receiver called Bob.

Step 1: embedding hidden messages ($\mathsf{Encode}_{\mathsf{ek}}^{\mathcal{C}_{\mathcal{H}}^{\lambda}}(m, \mathcal{H})$). As the most significant bit of a random $\mathbb{Z}_p$ element does not have uniform distribution (which is more biased to 0), to ensure (computational) indistinguishability between stegotext st and innocuous random elements $(c_j, r_j) \in \mathbb{Z}_p$, Alice embeds her secret message m in the least significant 252 bits of c_j and r_j, whereas, the most significant bits b_1 and b_2 are sampled according to the real distribution of c_j and r_j. As depicted in Fig. 3a, the rest consists of a 128-bit IV, 124-bit $\mathsf{Payload\ 1}$, and 252-bit $\mathsf{Payload\ 2}$. Let $F : \{0,1\}^{128} \times \{0,1\}^{128} \mapsto \{0,1\}^{128}$ be a PRF that takes as input a 128-bit plaintext and a 128-bit key, and outputs a 128-bit (pseudo-random) ciphertext. Alice uses synthetic IV to allow Bob to efficiently identify which transactions on the blockchain contain stegotext st. In particular, $\mathsf{IV} := F_z(\mathsf{rand} \| 00\ldots0)$, where $\mathsf{rand} \in \{0,1\}^{64}$ is a 64-bits random string, and $00\ldots0$ is a 64-bit string of 0's. As a result, to check if a signature contains any st, Bob can simply try to decrypt a suspected IV, obtaining $d := F_z^{-1}(\mathsf{IV})$. If the lower half of d consists of 64 bits of 0's, then this signature contains stegotext st.

In our attack, $\mathsf{Payload\ 1}$ and $\mathsf{Payload\ 2}$ are jointly used to convey a 376-bit hidden message m. The payloads are encrypted via a semantically secure symmetric encryption under the secret key z and using IV. Also, to handle an arbitrary-length hidden message and ensure the resulting ciphertext has the same length as the message (besides the IV), Alice can use the *Ciphertext Stealing* technique with CTR mode.



(a) CryptoNote: Format of one pair of ring signature random numbers (c_j, r_j) with 47-byte embedded stegotext.

(b) Bytecoin: embedding a 16-byte st in one pair of (c_j, r_j).

Fig. 3: Pair of subverted random numbers.

Step 2: identifying stegotext. Unlike conventional P2P covert communication, before attempting to extract a hidden message from a transaction, Bob should first identify if the target transaction contains a stegotext st. As mentioned before, Bob can accomplish this by parsing IV from the first two c_j's of the ring signature σ in a transaction, and checking whether the decryption of IV contains pattern 64 bits of 0's as shown in Fig. 3a. Note that Encode embeds the hidden message m in one of the first two pairs of (c_j, r_j). If c_1 does not yield the IV, then Alice's secret index ℓ must be 1, and Bob moves on to decrypt c_2 which must contain the IV, otherwise, the signature is an innocent cover text ct that does not contain st.

Step 3: extracting hidden messages ($\mathsf{Decode}_{\mathsf{dk}}(\mathsf{st})$). Once a steganographic ring signature is successfully identified, Bob can use the Decode algorithm to extract the hidden message. More specifically, Bob collects $\mathsf{Payload\ 1}$ and $\mathsf{Payload\ 2}$ as depicted in Fig. 3a. Bob then uses the extraction key $\mathsf{dk} := z$ to decrypt the payload, obtaining $m := \mathsf{Dec}_z(\mathsf{IV}, \mathsf{Payload\ 1} \| \mathsf{Payload\ 2})$.

Security. The proposed generic stegosystem against all CyptoNote-based cryptocurrencies is undetectable by Theorem 1. We remark that the content-insertion techniques that use non-standard Bitcoin scripts or exchange the public key with an arbitrary string with *printable* characters, as mentioned in [28] [1], can be detected. On the contrary, our proposed steganographic attack on CryptoNote simply replaces random numbers with pseudo-random ciphers which, by definition of semantic security, are computationally indistinguishable from each other.

Theorem 1. *The stegosystem $\mathcal{ST}$ for CryptoNote-based cryptocurrencies is IND-CHA secure, if $F : \{0,1\}^{\ell} \times \{0,1\}^{\lambda} \mapsto \{0,1\}^{\ell}$ is a secure PRF.*

Robustness and Efficiency. In terms of robustness, it is easy to see that, unlike image steganography, the stegotext embedded in the signatures can never be removed while still preserving the functionality of the signatures. Therefore, there is no filter that can remove our stegotext.

Throughput. The only similar attack in literature is the proof-of-concept attack in [6] which sends a hidden message bit-by-bit through the rejection-sampling of the transaction address. Besides sending one bit of the hidden message m per trans-

Demo Steganographic Bytecoin Transaction

- **Block height:** 1671177
- **Transaction id:** 52caba6ef4e4716ac8a25681eb3f380d3d1fee057ada7eb62d687af36f1a44ff
- **Sender's address:** 26c6YmZmLJZYxnVAt56kRraBhxiEUt8yoJR3VV4UV5VcRM9Pzs5qV7KStQHaa7xkAHej3WTTxtAc1KHbCSPoZ2ms3bdUsY6.
- **Receiver's address:** 26c6YmZmLJZYxnVAt56kRraBhxiEUt8yoJR3VV4UV5VcRM9Pzs5qV7KStQHaa7xkAHej3WTTxtAc1KHbCSPoZ2ms3bdUsY6.
- **Mixin count (ring size):** 6

Fig. 4: Stegangraphically-generated bytecoin transaction

Steganographic Monero Transaction

- **Block height:** 1502164
- **Transaction id:** e4b7982b081a17892525f1b1d3011ec06a0820cbf451d3a64f8ea998104a753c
- **Sender's address:** 455Bu1zXzgXEeXxrjzRSsEifP8WgtLTKYLreQ7RrA1fcFi2UKjgtc2UBapB9AcDaitdY7SdWGFsEZRELL8A1nMnEFRVZg47.
- **Receiver's address:** 42F5itWciYAg5QJxZEqWz5hrQNFaySUbxfxsjcdp8FnrRM68c8Nzujm3UqfscVC6r2c2GwuiP4sRsQv3ZZUc1spjUHuDHsx.
- **Mixin count (ring size):** 5

Fig. 5: A steganographically-generated Monero transaction

action, their attack also sends one transaction per block. As a result, with 10 minutes to add a new block in Bitcoin, a sender needs more than 24 hours to send a message of 20 bytes. On the other hand, our steganographic attack takes advantage of the randomness within each ring signature in CryptoNote transactions. In fact, a CryptoNote transaction contains a ring signature for each input. Therefore, if a transaction tx has y number of inputs, and k public keys in the ring of each signature, then the total number N of random numbers (c_j, r_j) in tx is $N = y*(k-1)*2$. Whereas, the available bandwidth B in bytes is $B = 32N$. Hence, the available bandwidth in one transaction of 10 inputs and 10 public keys is more than 5KB. In comparison, other techniques that replace segments of the transaction, e.g. replacing P2SH scripts in Bitcoin transactions as done in Tithonus [29], can at maximum transmit 1KB per transaction. Note that many blockchains offer an API to retrieve certain transactions and blocks. Therefore, if the receiver knows the heights, i.e. indices, of the blocks that contain the steganographically communicated data, he does not need to check the whole blockchain.

Robustness against blocking. Sometimes, censors can discover censorship-resistant proxies, e.g. Tor bridges, and block them. On the other hand, censors can not distinguish steganographically-subverted blockchain transactions; hence, they can not launch any targeted DoS attack unless they blacklist the whole blockchain which might have other financial ramifications. Additionally, from an attacker perspective, exploiting uncontrolled randomness is advantageous over other content-insertion approaches that simply replace segments of the transactions, as done in Tithonus [29] and Catena [30]. Namely, other techniques are susceptible to policy changes where certain scripts become conspicuous or no longer accepted, forcing the adoption of alternative techniques.

Cost. Content-insertion through the use of OP_RETURN transactions and the arbitrary replacement of transaction addresses [28] render the funds unspendable. Therefore, these techniques burns funds. On the contrary, our proposed steganographic attack does not incur any additional cost, except for minimal transaction fees, as the sender can always send transactions to his own addresses. Technically, we can choose arbitrarily large ring size in a transaction. In practice, however, we found that a value between 20 and 30 is the optimal ring size to get a transaction included quickly with minimum transaction fees. To further clarify the cost per Byte, a Bytecoin transaction tx with 2 inputs and 21 public keys can take about 2KB of covert data and costs 0.1 BCN as the minimum transaction fee which, given the current price of Bytecoin is \$0.000931 [12], costs \$0.0000931. Therefore, the cost of transmitting 1MB covertly is about \$0.05. Conversely, as shown in [29], Bitcoin-based Tithonus can covertly transmit up to 1650 Bytes in one transaction by replacing segments of the P2SH script of a multisignature transaction. Assuming the minimum transaction fee of 1 Satoshi/Byte and \$7925.83 [12] per Bitcoin, the cost of transmitting 1MB is more than \$83.

IV. Case studies: Bytecoin and Monero

In this section we show our implementation of the proposed attack in Sec. III in two *real* cryptocurrencies – Bytecoin and Monero. Specifically, we implemented the steganographic attack in Bytecoin (v 3.3.3)[2]. Similarly, we implemented and tested the attack in Monero. It is important to note that as of October 2018, Monero (v 0.13.0.0) has replaced Borromean ring signatures, that is exploited by our attack, by a succinct zero-knowledge proof called Bulletproofs, which is not covered by this work. Consequently, all of our discussion in relation with Monero is regarding v 0.12.0.0 and older.

Implementation in Bytecoin. Bytecoin is an open-source cryptocurrency project [31] that closely follows the CryptoNote protocol, which, as described in Sec. II, has sufficiently many uncontrolled random numbers that could be exploited to covertly communicate arbitrary information. Therefore, Bytecoin is susceptible to the generic steganographic attack described in Sec. III. Note that AES128 is used in the stegosystem because it is already implemented in Bytecoin source code.

As a proof-of-concept experiment and due to ethical reasons, we only covertly transfer 16 bytes in the real-world Bytecoin without significantly abusing the blockchain system. To generate steganographically hidden messages, the Bytecoin wallet's source code is changed by mainly modifying one source file, crypto.cpp, to alter the random numbers in the transaction's ring signature(s) and produce *one* pair of (c_j, r_j) as in Fig. 3b. Note that $j \neq \ell$ where ℓ is the signer's *secret*

[2] Although we describe the attack in (v 3.3.3), we have also successfully implemented the same attack in the most recent release (v 3.5.1), and the description is applicable with slight modifications.

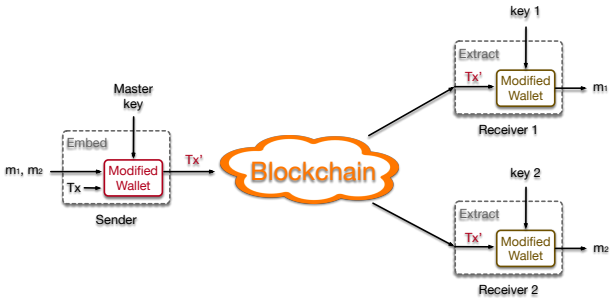


Fig. 6: Attack scenario 1: Covert broadcast communication.

index within the ring. Particularly, the changes introduced to crypto.cpp affect the following two functions within the source file: generate_ring_signature() and random_scalar().

To distinguish and identify signatures containing stego-text, function add_transaction() in BlockChainState.cpp is slightly modified to check each signature by decrypting each pair of (c_j, r_j) numbers. If a pattern is identified, it decrypts the most significant 16 bytes of r_j to extract m. Fig. 4 shows a demo subverted transaction included in the block at height 1671177 that contains a 16-byte hidden message "steganography".

Implementation in Monero (version 0.12.0.0). Although Monero is based on CryptoNote protocol, it uses Borromean ring signature which is different from the ring signature used in CryptoNote protocol as previously shown in Sec. II. Nevertheless, our generic attack in Sec. III is still applicable to Monero, which illustrates how the same attack can be extended to all public blockchain applications with randomized cryptographic primitives.

Monero has a very complex cryptographic structure and ring signature scheme in particular. The core of Monero's wallet involves Multilayered Linkable Spontaneous Anonymous Group Signature (MLSAG) and Borromean ring signature [32]. MLSAG is similar to the 1-out-of-n ring signature that is used as part of the CryptoNote protocol; however, rather than using a ring signature on a set of n keys, MLSAG uses a ring signature on a set of n-key vectors. Using MLSAG, the signer proves that he knows all the private keys corresponding to one column in the public keys' matrix. Also, despite the massive one-time secret key, the long-term secret key is still a single group element in $\mathbb{Z}_p$. Moreover, Borromean ring signature [32], which is a generalization and based on the 1-out-of-n signature [33], is used to mask the transferred amount while enabling the receiver to know how much they have received by revealing the mask [34].

In our experiment, we chose to exploit the Borromean ring signature as it offers higher throughput. Our attack on Monero is based on embedding a 32-byte hidden message m in the randomly generated $s_{i,j}$ numbers as part of the Borromean ring signature [32]. In particular, two vectors of $s_{i,j}$ numbers are generated by the genBorromean() function: $s_{0,j}$ and $s_{1,j}$. $s_{0,j}$'s are randomly generated when the j^{th} bit commitment is 1. Two of these randomly generated $s_{0,j}$'s are used to embed m. In a similar manner to our attack on Bytecoin,

Throughput and Cost					
Tool	**BW (B/Tx)**[(1)]	**Tx Fee (coin)**	**Price/coin**	**Tx Fee ($)**	**Cost 1 MB($)**
Our technique	2 KB[(2)]	0.1 BCN	0.000931	0.0000931	≈ 0.05
Tithonus	1650 B	10^{-8} BTC/B	7925.83	0.131	≈ 83
R3C3	1168 B	0.0001 ZEC	75.62	0.0076	≈ 6.8

TABLE II: Comparison between our covert broadcast technique, Tithonus [29] and R3C3 [36] in terms of throughput per transaction (BW) and cost of transmitting 1MB. Assuming a Bytecoin transaction with 4 inputs and ring size 10.

we use AES because it is already available in the source code. This step of the attack is achieved by slightly modifying two functions: genBorromean() and skGen() in two files: rctSig.cpp and rctOps.cpp. genBorromean() is modified to pass two extra parameters to skGen(). Also, blockchain.cpp is modified to check new transactions for steganographically hidden patterns, identify, and extract hidden messages.[3]

V. Attack Scenarios

This section describes the following attack scenarios: (i) covert broadcast communication, (ii) covert persistent storage, and (iii) wallet subversion attacks.

Attack Scenario 1: Covert Broadcast Channel. Conventional steganographic techniques typically assume that the covert communication is between two parties – a sender and a receiver. However, our steganographic attack can be used as a covert broadcast channel, i.e. one sender and multiple receivers. As analyzed in Sec. III, to steganographically send a hidden message of 1KB, Alice can easily craft a transaction with 4 inputs and 5 public keys. As shown in Fig. 6, our covert broadcast system utilizes our steganographic technique in conjunction with Boneh et al.'s broadcast encryption [11]. Our implementation is based on modifying the Bytecoin wallet (vr.3.1.1) using a tweaked version of the C implementation by Günther [35]. More specifically, we implemented a broadcast channel with up to 64 subscribers and one master node. As explained in Sec. III, the available bandwidth for steganographic information in one CryptoNote transaction is $BW = 32N$, where $N = y * (k - 1) * 2$, y denotes number of inputs, and k denotes the number public keys in each ring signature. Hence, in one transaction of 4 inputs and 10 public keys, our broadcast system can transmit more than 2KB of covert data.[4] Moreover, the minimum transaction fee is 0.01 BCN which costs $0.0000931. Therefore, the cost of transmitting 1MB data via this implementation and persistently storing it on the Bytecoin blockchain is approximately $0.05. Table II compares our implementation with Tithonous [29] and R3C3 [36], and shows that transmitting 1MB through Tithonous costs approximately $83, and approximately $6.8 in R3C3. [5]

The feasibility of this scenario and the high throughput demonstrate the severity of this attack, especially if abused

[3] More and specific details of the actual implementation in Bytecoin and Monero can be made available in the full version upon request.

[4] Boneh et al. broadcast encryption [11] sends a packet denoted as Hdr to all subscribers to enable them to derive a new symmetric encryption every time a new subscriber is added/removed. The size of Hdr is square-root of the number of users; when we have 64 users, Hdr consists of 9 elements. Since each element is 128 Bytes, the size of Hdr is 1152 Bytes.

[5] Comparison is based on prices of relevant cryptocurrencies quoted on 20/05/2019 from [12].

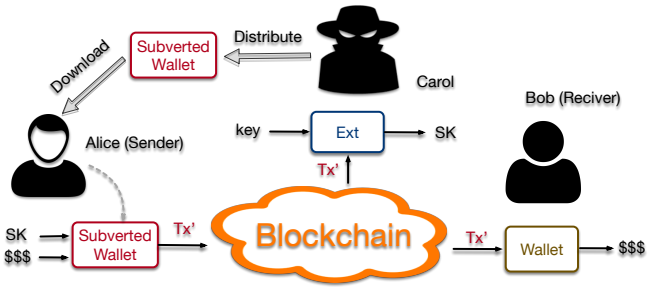


Fig. 7: Attach scenario 3: Subversion attack on crypto wallets to steal users' private keys

by outlaws to use public blockchains as covert broadcast networks for their illicit communication.

Attack Scenario 2: Covert Data Storage and Dissemination. Data storage can be viewed as a communication channel between the user and the user himself in the future. Unlike covert communication, covert persistent storage requires the uploaded content to be permanently stored and available on the blockchain. As aforementioned, the cost of covertly storing 1MB in Bytecoin's blockchain is about $0.05. Consequently, an adversary can use Bytecoin as a *cyberlocker* and abuse the P2P network of Bytecoin as a persistent content-delivery network (CDN). For example, it could be used to store pirated movies, wikileaks documents, etc. Another malicious example is for the attacker to covertly store private information about a victim for *blackmail*.

Attack Scenario 3: Wallet Subversion. In the aforementioned attack scenarios, the sender, Alice, is complicit in the malicious attacks. The third scenario depends on *different assumptions*, and presents a different situation where the sender is oblivious and is in fact a victim of the attack. Although this scenario may be applicable to *open-source* blockchain applications due to their complexity, it is more pertinent to *close-source* and hardware-based applications, e.g. hardware wallets. The significance of this attack scenario stems from its undetectability in the black-box setting, where secrets are leaked via normal transactions posted on the blockchain, and its serious repercussions on the victim's privacy and funds.

As depicted in Fig. 7, in this scenario, Alice is an innocent user who has downloaded, or bought, a wallet that is produced by a third party Carol who has maliciously implemented the wallet. In particular, Carol used a *subversion attack* to modify a wallet and redistribute it so to leak the signer's private key, while evading detection in the black-box setting. The way in which Carol modifies the wallet depends on the used cryptographic primitives and signature algorithms.

Below we present three subversion attacks that realize the scenario in Fig. 7. The first is a direct application of the generic steganographic attack described in Sec. III and its demo implementation in Bytecoin and Monero. Additionally, we present two more wallet subversion attacks targeting ECDSA-signature cryptocurrencies. The first attack on ECDSA-signature crypto wallets uses synthetic ephemeral key to covertly leak the entire signer's secret key over two signatures. However, it requires that the wallet is stateful in

Subverted ECDSA signing algorithm 1

$\mathsf{Sign}^{(1)}(\{S\}, s, m_1)$:
- Pick random $r_1 \leftarrow \mathbb{Z}_q$;
- Set $R' := g^{r_1} = (R'_x, R'_y)$;
- Set $w_1 := (\mathsf{hash}_p(m_1) + s \cdot R'_x) \cdot r_1^{-1} \pmod q$;
- Output $\sigma_1 := (R'_x, w_1)$;

$\mathsf{Sign}^{(2)}(\{S\}, s, m_2, r_1)$:
- Set $r_2 := \mathsf{hash}_p(Z^{r_1})$;
- Set $R := g^{r_2} = (R_x, R_y)$;
- Set $w_2 := (\mathsf{hash}_p(m_2) + s \cdot R_x) \cdot r_2^{-1} \pmod q$;
- Output $\sigma_2 := (R_x, w_2)$;

$\mathsf{Recover}(\sigma_1, \sigma_2, m_2, z)$:
- Set $R' \leftarrow \mathsf{map}(R'_x)$;
- Set $r'_2 := \mathsf{hash}_p((R')^z)$;
- Set $R := g^{r'_2} = (R_x, R_y)$;
- Output $s := (w_2 \cdot r'_2 - \mathsf{hash}_p(m_2)) \cdot (R_x)^{-1}$;

Fig. 8: The subverted ECDSA signing algorithm 1.

the sense that the wallet needs to store some variables from the previous signing execution. The second attack on ECDSA-signature crypto wallets is stateless and has lower throughput compared to the stateful attack.

Subverting ECDSA: Synthetic Randomness. Our first proposed subversion attack on ECDSA is a simplified version of the attack proposed in [37]. The subverted algorithm is depicted in Fig. 8. Let $z \in \mathbb{Z}_p$ be the adversary's secret key, and set $Z := g^z$. Let $R \leftarrow \mathsf{map}(R_x)$ be a mapping function that takes as input the x-coordinate and outputs the corresponding point on the curve. The subverted wallet needs to use algorithms $\mathsf{Sign}^{(1)}$ and $\mathsf{Sign}^{(2)}$ in turn to leak the signing key s. For the first time, $\mathsf{Sign}^{(1)}$ is identical to the original signature algorithm; however, the subtle difference is that $\mathsf{Sign}^{(1)}$ stores the ephemeral key r_1 in a long-term memory, which can be accessed during the next signature invocation. $\mathsf{Sign}^{(2)}$ is also similar to the original signature algorithm except that it deterministically generates $r_2 := \mathsf{hash}_p(Z^{r_1})$, where Z is hardcoded in the wallet. Once the adversary obtains two signatures σ_1, σ_2, he can use his secret key z to recover the victim's signing key s. First, he parses σ_1, σ_2 as (R'_x, w_1) and (R_x, w_2). The attacker then finds the point on the curve that corresponds to R'_x, using $R' \leftarrow \mathsf{map}(R'_x)$. After that, the attacker computes $r'_2 := \mathsf{hash}_p((R')^z)$. Note that if r'_2 is equal to r_2 then everything is correct. Let $R := g^{r'_2} = (R_x, R_y)$. The secret key can be extracted as $s := (w_2 \cdot r'_2 - \mathsf{hash}_p(m_2)) \cdot (R_x)^{-1}$. This attack illustrates how the entire long term signing key s can be leaked exclusively to the adversary over two subverted signatures.

Subverting ECDSA: Rejection Sampling. While our first ECDSA subversion attack has a very high throughput, it has few drawbacks. First of all, it is a stateful algorithm, so it is not suitable for all scenarios, especially for software wallets. Furthermore, the first attack can only leak the signing key by the nature of its design, and not any other confidential information. Note that most cryptocurrency wallets are able to avoid the re-use of the address and signing key. As a result, the leaked signing key in our first attack, may never

Subverted ECDSA signing algorithm 2

$\mathsf{Sign}(\{S\}, s, m_i, x)$:

- Repeat the following process:
  - Pick random $r \leftarrow \mathbb{Z}_q$;
  - Set $R := g^r = (R_x, R_y)$;
  - Compute $(j, b) \leftarrow \mathsf{PRF}_z(R_x)$;
  - If $x[j] = b$, break the loop;
- Set $w := (\mathsf{hash}_p(m_i) + s \cdot R_x) \cdot r^{-1} \pmod q$;
- Output $\sigma_i := (R_x, w)$;

$\mathsf{Recover}(\sigma_1, \ldots, \sigma_\ell, z)$:

- Init an array $\mathcal{S} := \emptyset$;
- For $i \in [\ell]$, do:
  - Parse σ_i as (u_i, v_i);
  - Compute $(j_i, b_i) \leftarrow \mathsf{PRF}_z(u_i)$;
  - Set $\mathcal{S}[j_i] := b_i$;
- Output $\mathcal{S}$;

Fig. 9: The subverted ECDSA signing algorithm 2

be used again even if the signing algorithms are executed twice with the same signing key. Nevertheless, for most wallets, there is a master key that is used to deterministically derive all the one-time signing keys.

As a result, our second subversion attack on ECDSA is stateless and is designed to leak arbitrary confidential information. As depicted in Fig. 9, the subverted signing algorithm takes as input the signing key s, the message m_i, and the secret $x \in \{0,1\}^n$ to be leaked. The signing algorithm leaks a random bit of x per signature. Let $\mathsf{PRF} : \{0,1\}^* \times \{0,1\}^\lambda \mapsto \{0,1\}^{\log n} \times \{0,1\}$ be a pseudo-random function that takes as input an arbitrary length message and the λ-bit PRF key, and it outputs a random number of $(\log n + 1)$ bits. The first $\log n$ bits is interpreted as an index j, and the last 1 bit is viewed as b. The subverted signing algorithm performs a rejection-sampling to find a random $R = (R_x, R_y)$ such that $(j, b) \leftarrow \mathsf{PRF}_z(R_x)$ and $x[j] = b$. The rest signing process is identical to the original signature algorithm. Note that the rejection-sampling is efficient, and the expected repetition per signature is 1.5 times.

To recover the secret, the adversary needs to obtain a collection of the signatures generated by the subverted algorithm. We emphasize that when the secret is a master key that can be tested for its correctness, it is *not* necessary to leak the entire key in practice. Assuming the master key is 256 bits, to obtain 50% distinct key bits, the expected number of signatures is bounded by approximately 256 signatures. Asymptotically, to obtain n secret bits, we need $\theta(n \log n)$ signatures. In practice, for a 128-bit key, we need about 179 signatures.

VI. Countermeasures

In a blockchain steganographic attack, the attacker is the sender; therefore, he/she can arbitrarily modify their client software to produce subverted transactions. This means that all randomized algorithms can be maliciously tampered with to embed secret information. In theory, deterministic cryptographic primitives can be used to eliminate the issue of uncontrolled randomness. However, most deterministic signature schemes, e.g. EdDSA, use synthetic randomness which

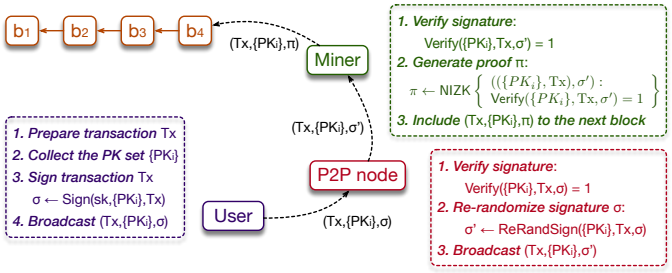


Fig. 10: Stego-resistant blockchain framework

is not a practical countermeasure against steganography; as the attacker can abuse or simply bypass the randomness generation, and it is impossible to verify an EdDSA signature is produced properly.

A Stego-Resistant Blockchain Framework (SRBF). A typical blockchain transaction contains one or more cryptographic components, such as signatures and non-interactive zero-knowledge (NIZK) proofs. We propose a universal stego-resistant blockchain framework that can be deployed to any blockchain system. Without loss of generality, we explain our technique to specifically defend against attacks on signature schemes; however, it can be applied analogously to NIZK proofs . As depicted in Fig. 10, the proposed SRBF introduces two elements to immunize blockchains against steganography and assumes that a given sender of any transaction may be maliciously implemented to exploit the random cryptographic signatures.

The first element of SRBF is to require P2P nodes to sanitize and re-randomize signatures. Currently, the P2P nodes in blockchains, e.g. in Bitcoin [38], check the validity of broadcast transactions and their signatures before relaying them into the network. In SRBF, P2P nodes also re-randomize the received signatures before propagating broadcast transactions. This practice filters out any possible steganographically embedded data and necessitates the use of re-randomizable signatures like Pointcheval's randomizable signature [39].

The second element of SRBF requires miners to replace signatures with NIZK proofs. Conventionally, upon receiving a transaction tx, the miners would check the validity of its associated signature σ, using the signature verification algorithm $\mathsf{Verify}(\mathrm{PK}, \mathsf{tx}, \sigma) = 1$. The miners then include the transaction together with its signature as it is to the next block, which will be eventually appended to the blockchain. However, in SRBF, the miner, instead of showing the signature, replaces the transaction's signature with a NIZK proof. Informally, the proof states that "I have seen a valid signature such that $\mathsf{Verify}(\mathrm{PK}, \mathsf{tx}, \sigma) = 1$". More precisely, we have

$$\mathcal{R}_{\mathsf{sig}} = \{((\{\mathrm{PK}_i\}_{i=1}^n, \mathsf{tx}), \sigma) | \mathsf{Verify}(\{\mathrm{PK}_i\}_{i=1}^n, \mathsf{tx}, \sigma) = 1\}$$

Thus, in SRBF, only $(\mathsf{tx}, \{\mathrm{PK}_i\}_{i=1}^n, \pi)$ will be posted on the blockchain, where:

$$\pi \leftarrow \mathsf{NIZK}\left\{ \begin{array}{l} ((\{\mathrm{PK}_i\}_{i=1}^n, \mathsf{tx}), \sigma) : \\ \mathsf{Verify}(\{\mathrm{PK}_i\}_{i=1}^n, \mathsf{tx}, \sigma) = 1 \end{array} \right\}$$

The security guarantee of SRBF is obvious, as signature σ is the witness of the corresponding NIZK proof. By NIZK

definition, π does not leak any information about σ. Therefore, all the steganographic information hidden in the signatures are filtered out from the blockchain. In practice, we can use Bulletproofs [18] as the NIZK instantiation. While our countermeasure slightly increases the computational effort for miners, it drastically decreases the effort for others to verify the validity of a given block; hence, mitigating the verifier's dilemma issue [40] when multiple transactions are combined in one NIZK proof. Note that, as a more efficient alternative to NIZK, miners can use aggregateable signatures, e.g. [41], to obfuscate single signatures and obliterate possible malicious content.

VII. Related Work

This work is related to the following topics.

Arbitrary content insertion in blockchains. was discussed in [28] where it is reported that 0.8% of 146 million Bitcoin transactions store content on the blockchain or use non-standard scripts. Similarly, the authors of [1] surveyed the methods that are used to store non-financial content, found that 1.4% of all Bitcoin transactions contain non-financial data, and retrieved over 1600 files, some of which contain objectionable content. Nonetheless, there are some benign applications that rely on content insertion in Bitcoin, like Tithonus [29] and Catena [30]. Also Minaei et al. [36] presented a Zcash-based censorship-bootstrapping tool, and explored content insertion techniques in Bitcoin, Zcash, Monero, and Etheruem.The authors of [6] are the first to discuss the use of *steganography* to *covertly* communicate in blockchains; however, they consider their attack to be a proof of concept and not practical due to its limitations.

Steganography. was introduced by Simmons' *prisoner's problem* [42]. Anderson et al. listed some of the limits of steganography and discussed the difficulty associated with formalizing a general proof of security for steganography [43] [44]. A number of works, e.g. [45]–[47], provided information-theoretic treatment of steganography security and robustness. More recently, Hopper et al. presented a definition for the security of a steganographic system in terms of the *computational* indistinguishability of stegotext from cover text [21].

Kleptography and Algorithm-Substitution Attacks. Our wallet subversion attack falls within the realm of *ASA* [26] [27], also called *Kleptography* [23] [24] and *Subversion Attacks* (SA) [48]. The notion of Kleptography was introduced by Young and Yung in 1996 [23] [24]. Subsequent work demonstrated the possible use of ASA in mass surveillance, and the susceptibility of all randomized symmetric encryption schemes to such attacks [27] [49]. Another demonstration of ASA attacks against the SSL/TLS and SSH2 protocols was presented in the work of Goh et al. [50]. In the context of signature schemes, Young and Yung [25] showed that DSA signature schemes can be subverted to leak secret information. In addition, Teşeleanu [51] described a threshold kleptographic attack on the generalized ElGamal signature that can be extended to similar DL-based signatures. Moreover, as a countermeasure against subversion, Russell et al. [52] modeled and proved a full domain hash-based signature scheme achieves subversion resilience. Other countermeasures include the work of Russell et al. [53] [54] who proposed a splitting-randomness technique to secure a randomizable IND-CPA public-key encryption, Ateniese et al. [48] who proposed the use of trusted reverse firewalls to re-randomize the output of signature algorithms, and Fischlin and Mazaheri [55] who proposed to proactively defend against ASA's assuming *temporary initial trust* of the possibly subverted algorithm.

Experiments on Bytecoin Wallet				
n	$\ell = k-1$ **(%)**	$\ell = k-2$ **(%)**	$\ell = k-3$ **(%)**	$\ell \geqq k-3$ **(%)**
3	86.61	12.85	0.54	100
4	81.11	17.39	1.45	99.95
5	76.82	20.61	2.45	99.88
6	70.23	25.28	4.15	99.66
7	66.43	27.64	5.31	99.38
8	61.2	30.48	7.17	98.85
9	57.09	33.05	8.24	98.38
10	54.37	33.39	10.07	97.83

TABLE III: Experimenting with Bytecoin wallet (v 3.0.0 and earlier) with different ring sizes, and the percentages show when the secret index ℓ is either one of the last indices in the ring.

VIII. Discovered ByteCoin Implementation Bug

CryptoNote-based cryptocurrencies use (linkable) ring signature to enhance the senders' privacy, i.e. set anonymity. As such, the signer's public key is hidden among a set of randomly sampled public keys. By its nature, the signer's public key should be indistinguishable from other keys in the ring. While experimenting with Bytecoin's wallet, we observed a major discrepancy between the wallet execution behaviour and the above-mentioned CryptoNote specifications. Peculiarly, contrary to the specifications that require the signer's public key to be randomly placed among the ring, Bytecoin tends to put the signer's public key as the last one in the ring. This bug nullifies the purpose of the ring signature.

Table III shows the statistical details of our experiments with Bytecoin wallet. The size of the ring k is set to a value between 3 and 10, and in each case, the function responsible for generating the ring signature, `generate_ring_signature(...)`, is invoked 10000 times. The table shows the percentages when the secret index ℓ is the last in the ring, i.e. when $\ell = k-1$, also when $\ell = k-2$ and $\ell = k-3$. It can be seen that the probability of $(\ell = k-1)$ increases as the size of the ring decreases. In addition, it can be observed that the probability is about 98% that the secret index ℓ is greater than or equal to $(k-3)$.

This bug effectively diminishes set anonymity promised by CryptoNote's ring signature, and facilitates blockchain analysis attacks. Hence, we reported the bug to Bytecoin's developers who replied to our report on 10 July 2018 and acknowledged the issue and their ongoing effort to rectify it. Note that this bug still exists in the latest version of Bytecoin (v 3.5.1) although the distribution of the secret index is biased towards the lower values, e.g. given n is 10, we found that $\ell \in \{0,1,2,3\}$ in $\geq 90\%$ of the time.

IX. Conclusion

The main aim of this work is to highlight the potential threat of maliciously abusing uncontrolled randomness in randomized cryptographic primitives in blockchain applications. To illustrate the idea, we designed, implemented, and evaluated our attacks against the widely-used ECDSA signature scheme, the ring signature used in the CryptoNote framework, and the Ring Confidential Transaction used in Monero (up to version 0.12.0.0). The demonstrated attacks can be used in three malicious scenarios: covert communication, persistent storage of objectionable data, and wallet subversion attacks. Finally, we emphasize that this line of research is far from being completed, and we hope that our work motivates the design of stego-resistant blockchains.

Acknowledgement

Bingsheng Zhang is supported by IOHK.

References

[1] R. Matzutt, J. Hiller, M. Henze, J. H. Ziegeldor, D. Mullman, O. Hohlfeld, and K. Wehrle, "A quantitative analysis of the impact of arbitrary blockchain content on bitcoin," in *FC 2018*, 2018.

[2] J. Smith, J. Tennison, P. Wells, J. Fawcett, and S. Harrison, "Applying blockchain technology in global data infrastructure," tech. rep., Open Data Institute, June 2016. ODI-TR-2016-001.

[3] R. Matzutt, M. Henze, J. H. Ziegeldorf, J. Hiller, and K. Wehrle, "Thwarting unwanted blockchain content insertion," in *IC2E 2018*, pp. 364–370, April 2018.

[4] I. Puddu and A. Dmitrienko, "μchain: How to forget without hard forks," *IACR Cryptology ePrint Archive 2017/106*, 2017. https://eprint.iacr.org/2017/106.

[5] G. Ateniese, B. Magri, D. Venturi, and E. Andrade, "Redactable blockchain – or – rewriting history in bitcoin and friends," in *Euro S&P 2017*, pp. 111–126, April 2017.

[6] J. Partala, "Provably secure covert communication on blockchain," *Cryptography*, vol. 2, no. 3, 2018.

[7] G. Fuchsbauer, "Subversion-zero-knowledge snarks," in *PKC 2018*, pp. 315–347, 2018.

[8] B. Abdolmaleki, K. Baghery, H. Lipmaa, and M. Zajac, "A subversion-resistant snark," in *ASIACRYPT 2017*, pp. 3–33, 2017.

[9] S. Azouvi, M. Maller, and S. Meiklejohn, "Egalitarian society or benevolent dictatorship: The state of cryptocurrency governance," in *5th Workshop on Bitcoin and Blockchain Research*, 2018.

[10] Giza Device Ltd, "Giza wallet," 2017. Avialable Online: https://www.gizadevice.com/ (Last accessed 7-Feb-2018).

[11] D. Boneh, C. Gentry, and B. Waters, "Collusion resistant broadcast encryption with short ciphertexts and private keys," in *Advances in Cryptology – CRYPTO 2005* (V. Shoup, ed.), (Berlin, Heidelberg), pp. 258–275, Springer Berlin Heidelberg, 2005.

[12] CoinMarketCap, "Cryptocurrency market capitalizations," 2018. Available Online: https://coinmarketcap.com/ (Last accessed 26-Nov-2018).

[13] N. V. Saberhagen, "Cryptonote v 2.0," 2013. whitepaper, Available online: https://cryptonote.org/whitepaper.pdf, (Last accessed 23-Nov-2018).

[14] Bytecoin Org., "Bytecoin (bcn)," 2018. Available Online: https://bytecoin.org/ (Last accessed 23-Nov-2018).

[15] CryptoNote Org., "Cryptonotecoin," 2018. Available Online: http://cryptonote-coin.org/ (Last accessed 23-Nov-2018).

[16] Fantomcoin, "Fantomcoin," 2014. Available Online: http://fantomcoin.org/ (Last accessed 23-Nov-2018).

[17] Monero, "Monero," 2018. Available Online: https://getmonero.org/ (Last accessed 07-Feb-2018).

[18] B. Bünz, J. Bootle, D. Boneh, A. Poelstra, P. Wuille, and G. Maxwell, "Bulletproofs: Short proofs for confidential transactions and more," in *S&P 2018*, vol. 00, pp. 319–338, 2018.

[19] S. Noether, "Ring signature confidential transactions for monero." Cryptology ePrint Archive, Report 2015/1098, 2015.

[20] T. P. Pedersen, "Non-interactive and information-theoretic secure verifiable secret sharing," in *CRYPTO '91*, pp. 129–140, 1992.

[21] N. J. Hopper, J. Langford, and L. von Ahn, "Provably secure steganography," in *CRYPTO 2002*, 2002.

[22] N. Dedić, G. Itkis, L. Reyzin, and S. Russell, "Upper and lower bounds on black-box steganography," *Journal of Cryptology*, vol. 22, pp. 365–394, Jul 2009.

[23] A. Young and M. Yung, "The dark side of "black-box" cryptography or: Should we trust capstone?," in *CRYPTO '96*, 1996.

[24] A. Young and M. Yung, "Kleptography: Using cryptography against cryptography," in *EUROCRYPT '97*, 1997.

[25] A. Young and M. Yung, "The prevalence of kleptographic attacks on discrete-log based cryptosystems," in *CRYPTO '97*, 1997.

[26] M. Bellare, K. G. Paterson, and P. Rogaway, "Security of symmetric encryption against mass surveillance," in *CRYPTO 2014*, (Berlin, Heidelberg), pp. 1–19, Springer Berlin Heidelberg, 2014.

[27] M. Bellare and J. Jaeger, "Mass-surveillance without the State : Strongly Undetectable Algorithm-Substitution Attacks," in *CCS*, 2015.

[28] R. Matzutt, O. Hohlfeld, M. Henze, R. Rawiel, J. H. Ziegeldorf, and K. Wehrle, "Poster: I don't want that content! on the risks of exploiting bitcoin's blockchain as a content store," in *CCS '16*, 2016.

[29] R. Recabarren and B. Carbunar, "Tithonus: A bitcoin based censorship resilient system," *Proceedings on Privacy Enhancing Technologies*, vol. 2019, no. 1, pp. 68 – 86, 2019.

[30] A. Tomescu and S. Devadas, "Catena: Efficient non-equivocation via bitcoin," in *2017 IEEE Symposium on Security and Privacy (SP)*, vol. 00, pp. 393–409, May 2017.

[31] B. D. Team, "Bytecoin project github repository," 2018. Available Online: https://github.com/bcndev (Last accessed 26-Nov-2018).

[32] G. Maxwell and A. Poelstra, "Borromean Ring Signatures," 2015. Available Online: http://diyhpl.us/~bryan/papers2/bitcoin/Borromean%20ring%20signatures.pdf (Last accessed 07-Feb-2018).

[33] M. Abe, M. Ohkubo, and K. Suzuki, "1-out-of-n signatures from a variety of keys," in *ASIACRYPT 2002*, 2002.

[34] G. Maxwell, "Confidential Transactions," 2018. Available Online: https://people.xiph.org/~greg/confidential_values.txt (Last accessed 07-Feb-2018).

[35] O. Günther, "Broadcast key encapsulation mechanism github repository," 2012. Available Online: https://github.com/oliverguenther/PBC_BKEM (Last accessed 08-May-2019).

[36] M. Minaei, P. Moreno-Sanchez, and A. Kate, "R3c3: Cryptographically secure censorship resistant rendezvous using cryptocurrencies." Cryptology ePrint Archive, Report 2018/454, 2018. Available online: https://eprint.iacr.org/2018/454 (Last accessed 14-Feb-2019).

[37] E. Mohamed and H. Elkamchouchi, "Kleptographic Attacks on Elliptic Curve Cryptosystems," *Journal of Computer Science*, vol. 10, no. 6, pp. 213–215, 2010.

[38] P. Koshy, D. Koshy, and P. McDaniel, "An analysis of anonymity in bitcoin using p2p network traffic," in *Financial Cryptography and Data Security* (N. Christin and R. Safavi-Naini, eds.), (Berlin, Heidelberg), pp. 469–485, Springer Berlin Heidelberg, 2014.

[39] D. Pointcheval and O. Sanders, "Short randomizable signatures," in *Topics in Cryptology - CT-RSA 2016* (K. Sako, ed.), (Cham), pp. 111–126, Springer International Publishing, 2016.

[40] L. Luu, J. Teutsch, R. Kulkarni, and P. Saxena, "Demystifying incentives in the consensus computer," in *Proceedings of the 22Nd ACM SIGSAC Conference on Computer and Communications Security*, CCS '15, (New York, NY, USA), pp. 706–719, ACM, 2015.

[41] D. Boneh, C. Gentry, B. Lynn, and H. Shacham, "Aggregate and verifiably encrypted signatures from bilinear maps," in *Advances in Cryptology — EUROCRYPT 2003* (E. Biham, ed.), (Berlin, Heidelberg), pp. 416–432, Springer Berlin Heidelberg, 2003.

[42] G. J. Simmons, *The Prisoners' Problem and the Subliminal Channel*, pp. 51–67. Boston, MA: Springer US, 1984.

[43] R. Anderson, "Stretching the limits of steganography," in *Information Hiding*, pp. 39–48, Springer Berlin Heidelberg, 1996.

[44] R. J. Anderson and F. A. P. Petitcolas, "On the limits of steganography," *IEEE Journal on Selected Areas in Communications*, vol. 16, pp. 474–481, May 1998.

[45] J. A. O'Sullivan, P. Moulin, and J. M. Ettinger, "Information theoretic analysis of steganography," in *Proceedings. 1998 IEEE International Symposium on Information Theory*, 1998.

[46] T. Mittelholzer, "An information-theoretic approach to steganography and watermarking," in *Information Hiding*, 2000.

[47] C. Cachin, "An information-theoretic model for steganography," in *Information Hiding*, 1998.
[48] G. Ateniese, B. Magri, and D. Venturi, "Subversion-resilient signature schemes," in *CCS '15*, pp. 364–375, 2015.
[49] M. Bellare and V. T. Hoang, "Resisting randomness subversion: Fast deterministic and hedged public-key encryption in the standard model," in *EUROCRYPT 2015*, 2015.
[50] E.-J. Goh, D. Boneh, B. Pinkas, and P. Golle, "The design and implementation of protocol-based hidden key recovery," in *Information Security*, 2003.
[51] G. Teseleanu, "Threshold kleptographic attacks on discrete logarithm based signatures." Cryptology ePrint Archive, Report 2017/953, 2017. https://eprint.iacr.org/2017/953.
[52] A. Russell, Q. Tang, M. Yung, and H.-S. Zhou, "Cliptography: Clipping the power of kleptographic attacks," in *ASIACRYPT*, 2016.
[53] A. Russell, Q. Tang, M. Yung, and H.-S. Zhou, "Destroying steganography via amalgamation: Kleptographically cpa secure public key encryption." Cryptology ePrint Archive, Report 2016/530, 2016.
[54] A. Russell, Q. Tang, M. Yung, and H.-S. Zhou, "Generic semantic security against a kleptographic adversary," in *CCS '17*, (New York, NY, USA), pp. 907–922, ACM, 2017.
[55] M. Fischlin and S. Mazaheri, "Self-guarding cryptographic protocols against algorithm substitution attacks," in *2018 IEEE 31st Computer Security Foundations Symposium (CSF)*, pp. 76–90, July 2018.