ChainChannels: Private Botnet Communication Over Public Blockchains

Davor Frkat
Institute of Telecommunications
TU Wien
Vienna, Austria
davor.frkat@tuwien.ac.at

Robert Annessi
Institute of Telecommunications
TU Wien
Vienna, Austria
robert.annessi@nt.tuwien.ac.at

Tanja Zseby
Institute of Telecommunications
TU Wien
Vienna, Austria
tanja.zseby@tuwien.ac.at

***Abstract*—Botnets provide the basis for a wide range of malicious activities in the Internet. Sophisticated Command & Control infrastructures aim to prevent the detection and takedown of botnets and therefore pose a big challenge in the battle against network attacks of all kinds. In this paper we present a method for covert botnet communication that exploits the digital signatures used in blockchains to inject subliminal messages. We show how subliminal messages can be included in signatures and distributed in blockchain transactions to the bots. We also show how keying material required for extracting the subliminal information can be transmitted privately to bots by being stored in a public blockchain. As proof of concept we injected a subliminal message and a key in the Bitcoin blockchain and show how this information can be extracted from the transaction.**

Our method allows to establish a hidden Command & Control infrastructure over blockchains and send instructions to all bots without any suspicious communication activities. The method relies only on digital signatures and is therefore applicable to numerous blockchains. The subliminal communication can not be distinguished from legitimate transactions and mitigation would require to redesign the blockchains to use new subliminal-free signature schemes. Our method provides a general hidden distribution channel over blockchains and can be also applied to other scenarios where information needs to be transmitted covertly. It scales extremely well with the number of receivers (here bots), and subliminal messages can even be distributed over different blockchains to exploit specific features of blockchains such as low transaction cost or fast confirmation times or to further obfuscate the existence of the C&C communication.

***Index Terms*—blockchain, botnet, network security, subliminal channel, digital signatures**

I. Introduction

Botnets provide very powerful infrastructures for various malicious activities in the Internet. The target of botnet operators is to produce an economically cheap, logistically feasible, hidden, fast, and robust C&C network, which is ideally impossible to take down or rather difficult to obstruct in its functions. In the past, the race between botnet developers and their adversaries, such as competing botnet operators or authorities, led to highly innovative and sophisticated command and control (C&C) infrastructures [1]. The main weak point and leverage against botnets often turned out to be a vulnerability in the C&C concept, which could be used for the detection and take-down of a botnet.

In this paper we present a novel way of privately broadcasting information over public blockchains that can be used for the distribution of C&C commands to associated bots. For this purpose, we include subliminal information in the digital signatures that are used to secure blockchain transactions. Since digital signatures are essential for the operation of blockchains, they provide a distributed transmission method that can be exploited by botnet operators.

We show how a secret sharing scheme can be used to distribute keying material (needed to decrypt the subliminal information) secretly to the bots and to avoid storing the private key in advance in a bot to prevent take-over by an adversary that acquired information from a compromised bot. An adversary can follow the communication with a compromised bot, but can not use it to take control over the botnet.

As proof of concept we injected a subliminal message in the Bitcoin blockchain and explain how the subliminal information can be extracted. We also implemented our method to leak the private key in the experiment so that the applicability of our method can be verified. Our method is not restricted to a specific blockchain and robust against takeover. Even if an observer has knowledge about the subliminal message transmission and the key, he cannot forge own commands and the size of botnet remains unknown.

Our contribution in this paper can be summarized as follows:

- We introduce a new method to establish a hidden C&C infrastructure for broadcasting information to bots by transmitting subliminal information in blockchain signatures. Our method uses broadband subliminal channels, is not restricted to a specific blockchain and can be even distributed over multiple blockchains.
- We show as an example how the method can be applied using the classical Elliptic Curve Digital Signature Algorithm (ECDSA), which is commonly used in many blockchains. Our method can also be applied if other signature schemes are used, including modern high speed signatures such as Edwards-curve Digital Signature Algorithm (EdDSA) and signatures schemes based on Multivariate Quadratic Quasigroups (MQQ),

- We present a method to secretly transmit keying material to the bots that enables them to decode the subliminal information without the need to store the private key in the initial bot configuration, and therefore reducing the trust required in the bots.
- We conducted an experiment as proof of concept and injected a subliminal message and a key into the Bitcoin blockchain. This message provides an example how commands can be transmitted covertly from a C&C server to all bots.
- We discuss several mitigation approaches. Nevertheless, all methods have severe drawbacks and are not applicable to current blockchains.

II. Background and Related Work

A. Botnet Communication

Botnets are a highly researched area and show a wide variation of communication protocols, patterns and infrastructure designs [1]. The main terms for botnets are:

- The botmaster, who controls the botnet and tries to stay stealthy.
- The bots reside on infected hosts and use the machine's resources to perform tasks as commanded by the botmaster, such as stealing private information or performing distributed denial of service attacks (DDoS).
- The Command and Control (C&C) infrastructure can be centralized, a peer to peer (P2P) structure or a hybrid. It is used to send commands to the bots. This can be as simple as Internet relay chat channels (IRC), HTTP, or more sophisticated neoteric protocols [1]. The C&C infrastructure is the key to the robustness of a botnet against takedown or takeover.
- Adversaries, i.e. other malicious individuals or the authorities who try to take down the botnet for different reasons.

B. Subliminal Channels in Signature Schemes

Simmons was the first to show how a narrowband of few bits or a broadband channel could be implemented in digital signatures [2, 3]. He introduced a model with two communicating parties, so-called prisoners, who are monitored by a warden. The prisoners are allowed to communicate and use signatures that are checked by the warden. Simmons shows that a subliminal message can be embedded in the signature while it is valid and the existence of the hidden message can not be detected.

Subliminal channels are not to be confused with steganography. Steganography uses other objects (such as images) to transmit information and often relies on security by obscurity. Subliminal channels on the other hand are employed in cryptographic schemes, whereby the resulting communication can not be distinguished from any other message not carrying the channel [4].

Subliminal channels are divided into two categories, narrowband and broadband subliminal channels. In the narrowband variation the signatures are calculated repeatedly with different values for the nonce, until a defined number of bits, e.g. the x first or last, coincide with a determined sequence of bits [5]. The main problem with this method is, that the computational effort rises with 2^x, which makes this channel viable for only a few bits. In contrast a broadband channel uses the complete size nonce to hide information. The main disadvantage is, that the private key of the signature is also needed by the receiver to decode the message. Since the same key is used to perform transactions on a blockchain, we will later introduce a method has to exchange the key, without a compromised receiver being able to forge his own transactions beforehand. The two main digital signature algorithms (DSAs) in use for blockchain applications are ECDSA and EdDSA, which are both shown to have broadband subliminal channels [5, 6]. Several further signature schemes that could be used in blockchain applications may also allow the injection of subliminal information [7, 8].

C. Data Insertion in Blockchains

Blockchains were initially used for cryptocurrencies, but more applications developed on top of them, such as time stamping and notary services but also malicious and illicit content [9]. The Bitcoin blockchain was extensively analyzed for arbitrary data [10, 11]. It was shown, that the OP_RETURN field, and payments to fake public key, key hashes or script hashes, in which the data is stored can be used to inject data. The coins from these addresses can not be claimed, since the private keys can not be derived. The previous authors show that as long as the transactions are valid, arbitrary content can be injected into public blockchains. Nevertheless, the content is transmitted overtly and can be observed by anyone.

Some previous work [12][13] already showed that the Bitcoin blockchain is viable as C&C infrastructure by using key leakage, the OP_RETURN field, and a narrowband subliminal channel based on brute-forcing the random factor of the signature scheme to set a number of arbitrary bits in the signature. In their papers the authors point out, that C&C communication over the Bitcoin Network inherits its key strengths. The authors argue, that a blockchain network used for botnets is offers low latency, is distributed and decentralized by design, has a consistent network state and it would be hard to censor C&C instructions without significantly impacting the function of the blockchain. Also the proposed communication channel scales well with the number of bots and offers the botmaster an infrastructure which is not likely to be taken down and thus less risky and less costly.

The method presented in the Zombiecoin paper [13] is specific for the Bitcoin blockchain. It uses the OP_RETURN field and only proposes the use of a narrowband subliminal channel, which is computationally expensive and not practical to use. In contrast, our paper is based on a key leakage method and a broadband subliminal channel, which is easy to calculate and where the complete random factor can be used to transmit data. Our method does not require any specific blockchain field. It relies solely on the one thing that all blockchains have in common - the signatures.

III. Building Blocks for C&C Communication

In this section we describe the building blocks required to establish a subliminal channel for C&C communication. In the following we often refer to the Bitcoin blockchain as example for a widely used blockchain application. Nevertheless, our methods are applicable to any other blockchain application as long as the blockchain uses appropriate signatures such as ECDSA that allow the injection of subliminal messages.

For the establishment of the C&C infrastructure we make the following assumptions:

1) We focus our contribution to the distribution of commands to the bots. We consider all prerequisites and communications (e.g., distribution of keying material) needed to establish an environment to transmit those commands as in scope. But the initial botnet establishment, i.e., the scanning for vulnerable hosts as potential new bots and the initial propagation of bot software to compromised hosts, can be done by a wide range of mechanisms (including the use of subliminal communication) and is not discussed here. Also any potential upstream communication from the bots to the botmaster is considered out of scope.
2) We assume that bots are connecting directly to the blockchain and extract the signatures from the corresponding fields (e.g., *ScriptSig* for Bitcoin). Alternatively, the bots can use blockchain explorer websites or their respective APIs (over HTTP or HTTPS). This can be implemented to reduce communication overhead or as a fallback option when bots do not have access to certain ports (e.g., tcp/udp 8333 for Bitcoin) due to firewall restrictions for example.
3) All bots are provided with at least one initial public key, public key hash or address on which they check for new commands. After a command is finished, they are provided with new information on where to look for subsequent instructions. Newly infected bots need to have the current address, since they would follow through all commands starting from the initial transaction.
4) All transactions are designed in a way, that no coins are left to claim after a private key has been exposed on purpose. For simplicity we assume only one input address per transaction.

The building blocks required to establish the subliminal channel for C&C infrastructure are the following:

1) A signature scheme such as ECDSA or EdDSA that provides the possibility to inject a subliminal channel.
2) A method to secretly exchange information required to use the subliminal channel, i.e. in our case this is the distribution of the private key to the bots.

Both building blocks are described in detail in the sections below.

IV. Subliminal Channels in Blockchain Signatures

As an example we use the public blockchain for the Bitcoin cryptocurrency. The Bitcoin blockchain uses the ECDSA signature scheme with the curve secp256k1[14]. Since there are many variations and developments stemming from Bitcoin, the introduced subliminal channel is not limited to the original Bitcoin blockchain and indoependent from other design decisions. For secp256k1 the initial parameters, the generator point G and order of the curve n are publicly known.

An ECDSA signature consists of the pair (r, s), which is calculated as follows:

1) A cryptographically secure nonce k is generated per message in the range of $1 \leq k \leq n-1$. Measures have to be taken so that k can not be guessed by an adversary, which are left to the implementor.
2) A new point on the curve is computed by $kG = (x_1, y_1)$. The first coordinate (x_1) is used as first part of the signature, i.e., $r = x_1$.
3) The message m to be signed consists of a raw preliminary transaction [15]. The message is hashed twice with SHA-256, refered to as hashed message z.
4) The second part of the signature s is then calculated from the private key d, the hashed message z, the parameter r and the multiplicative inverse k^{-1} of the nonce $k \mod n$ as follows:

$$s = k^{-1}(z + r \cdot d) \mod n \tag{1}$$

This leads to the complete signature pair (r, s), which is openly embedded in the final transaction

In order to insert a subliminal channel in the signature the random value k is filled with the subliminal message. The signature carrying the subliminal message is then generated the same way, just that k now contains the message instead of the random number.

In order to extract the subliminal information from the signature the receiver needs to know the private key d. The subliminal message k then can be extracted by the receiver as follows:

$$k = s^{-1} \cdot [z + (d \cdot r)] \mod n \tag{2}$$

The signature is valid independent of the specific value chosen for k. However it should be avoided to use repeating values of k, especially for the same private keys. If the same k is used multiple times (as it may be the case for command and control messages in botnets), this information can be used to leak the private key without intending it [16].

In the next section we show how the receiver can obtain the information about the private key d from blockchain operations and without the need to pre-configure the bot with the private key. Storing the private key into the bot would imperil the whole botnet. Instead, we implement mechanisms to provide the complete key when it is safe to assume that the embedded subliminal message can not be altered anymore.

The further discussion and notation is refering to the original Bitcoin blockchain and ECDSA as its corresponding signature scheme. Nevertheless, while ECDSA is a signature used in many blockchains, subliminal channels also exist in other signatures. An emerging high-speed signature scheme is EdDSA [17]. Especially the Ed25519 variation is used or

planned to be used in various blockchain applications [18]. As demonstrated in [5], a broadband subliminal channel in EdDSA can be established similar to the one in ECDSA. Also MQQ-based signature schemes allow the establishment of subliminal channels as demonstrated in [8]. Especially all the MQQ schemes that are currently considered secure are vulnerable to subliminal channels. Therefore, the basic idea and all the following concepts can easily be applied to blockchains using EdDSA or MQQ-based signature schemes. The same goes for Schnurr signatures [7].

V. Distribution of the Private Key

Subliminal messages are pushed by the sender to the blockchain (or are awaiting confirmation in the mempool) and the receiver knows from which address to expect the subliminal instructions. Nevertheless, as shown above the bots require knowledge about the private key d of the sender in order to extract the subliminal messages. In this section, we present different methods how the private key can be distributed.

Furthermore, in Section III we assumed that the receiver knows which address it needs to listen for. Nevertheless, with the three of the methods described below the bots can get information about the private key just by searching for some patterns in all transactions in the blockchain. Therefore they do not require an initially pre-configured sender address in the bots. The bots instead receive the private key and then can calculate the sender address from the private key as depicted in Fig. 1.

A. Pre-Configuration of the Private Key in All Bots

The easiest method would be to simply pre-configure the private key in the bots (e.g., as part of the bot code). An advantage of this method is that the bots do not need to have the address or public key pre-configured to know in which transactions the subliminal information is encoded. Knowing the private key is sufficient to deduce the public key and with this the sender address. Nevertheless, with this method an adversary (to the bot owner) just needs to compromise a single bot to gain knowledge of the private key. With this knowledge the adversary can take over the botnet since messages can be crafted that valid and indistinguishable from the actual bot owner.

B. Key Leakage Based on Nonce Reuse

A better method to provide the key informatin to the bots is to send it to them after the botnet is established. This way the key needs to be revealed only when the botnet is unlocked, e.g., shortly before the commands should be executed. Therefore it is also possible that the bots first just collect all the messages received from the sender without decoding them. And then later when they receive the private key start decoding the messages and trigger the execution of the commands. This way the key remains unknown to the bots (and undetectable for any adversaries) until botnet actions are triggered.

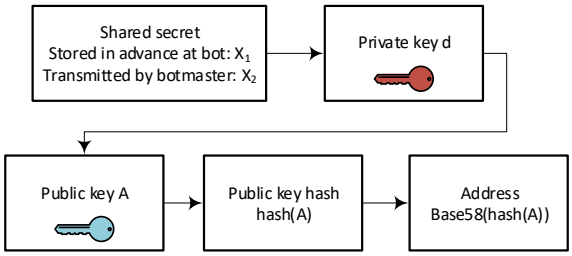

Fig. 1. From the view of a bot: Simplified relation of the shared secrets X_1 and X_2 to the key pair and the address. Everything can be derived with both parts of the shared secret.

The method to distribute the private key to the bots is to use a classical key leakage method based on nonce reuse. This is done by deliberately reusing the nonce k. Since r depends only on k and G this produces two signatures (r, s_1) and (r, s_2) with two different messages m_1 and m_2 but identical values of r. The calculation of the private key is presented in [16]. The receiving bots are looking for all transactions from the sender address and check the r values. When the second transaction (with a reused nonce) is passed, the bots are able to calculate the private key from the two signatures and can trigger the execution of commands.

Bad random number generators can generate identical k values by accident that lead to the same r values in distinct signatures. Anyone who finds such duplicates can discover the private key and can use it for instance to steal Bitcoins. Therefore, the detection of duplicates is a well established method and tools exist to search for identical r values in signatures. By using such tools over all transactions the bots can get the private key without pre-configured knowledge about the sender of the subliminal information. Instead, they can derive the source of the subliminal information after calculating the private key.

Nevertheless, one drawback of this key leakage method is that everyone could notice the two identical r values by parsing the blockchain. Hence anyone can derive the leaked private key. Subsequently, also the messages transmitted to the bots would be visible to anyone with access to the blockchain who knows or suspects that a subliminal channels is used. Therefore, it would be useful to distribute the private key in a way that only the bots can decode it. We show two approaches in the subsections below.

C. Secret Sharing

To solve the problem of exposing the C&C messages to the public, a secret sharing scheme is introduced. There exist various secret sharing schemes, such as Shamir's [19] or Brickell's [20].

For the problem at hand a plain secret sharing scheme is sufficient. It can be upgraded without affecting the broad principle of operation. For the simple secret sharing scheme

the secret private key d is split into two parts X_1 and X_2 by following the scheme in (3b) and (3c). For this a cryptographically secure random number R is needed. A large prime n is used for the modulo operation and defines the range of d and R as seen in (3a). Since the size of the key to be split up is 32 byte, the (prime) order n of the secp256k1 curve [14] can be used.

$$0 < d < n \text{ and } 0 < R < n \tag{3a}$$
$$X_1 = (d + R) \mod n \tag{3b}$$
$$X_2 = (d + 2R) \mod n \tag{3c}$$

The message can easily be reconstructed when both parts of the shared secret are combined:

$$d = (2 \cdot X_1 - X_2) \mod n \tag{4}$$

We now assume that all bots know the value X_1. The main difference to the key leakage based on nonce reuse is, that only bots know the common X_1 and the leaked information X_2 is useless without the common secret X_1 that the bots posses. Furthermore, the information leaked can be used to calculate the private key and subsequently the source address of the transaction as depicted in Fig. 1. The bots now store the pair of the shared secret (X_1) and the address (A). X_1 can simply be pre-configured in the bots code without the danger that an adversary gets direct information about the private key if a bot gets compromised. Also with this method bots can use tools to search for duplicate r values over all transactions. So they also can get the private key without pre-configured knowledge about the sender of the subliminal information and later derive the sender address from the private key.

One important detail with all secret sharing schemes is that the address from which the X_2 is leaked has to be different from the address from which the subliminal messages are sent. The reason for this is that with the leaked information the private key of the leaking address can be derived. So if there are two equal r values observed from address A_1, everyone can derive X_2 and therefore the private key for A_1, if the subliminal messages were also sent by A_1. If the subliminal messages are sent from a different address (e.g., A_2), however, then the X_1 value is required together with the X_2 to calculate the private key of A_2, and only bots that had the X_1 pe-configured can decode the subliminal messages.

D. Concealed Key Leakage

Until now, we have secured the transactions from being directly exposed to the public. The problem remains, that the existence of nonce reuse alone may raise suspicion especially given the fact that blockchains are constantly monitored, mainly to attempt to steal any coins associated with a leaked key [21]. To stay covert and to not produce suspicious transactions, we propose to use an advanced method and avoid using the same nonce k twice. Instead, we link the r part of one signature with a common secret, the shared secret X_1, which the bots already have, as seen in (5c). By using another nonce, the resulting r differs due to $r_1 = k_1 * G$, the private key can still be recovered by bots because they know the shared secret X_1 as well as r_1, which is public. With X_1 and r_1 they can derive the nonce k_2 used in tx_2 and therefore the private key. To an attacker the signature pairs (r_1, s_2) and (r_2, s_2) are completely indistinguishable from signatures of regular transactions. They neither trigger any key leakage detection nor raise any suspicion by observers. The bots, however, who know the shared secret can derive the private key used and therefore extract the subliminal information included in the signatures of the other transactions.

$$tx_1 = (r_1, s_1) \tag{5a}$$
$$tx_2 = (r_2, s_2) \tag{5b}$$
$$k_2 = r_1 \oplus X_1 \tag{5c}$$
$$X_2 = [(s_2 \cdot k_2) - z_2] \cdot r_2^{-1} \mod n \tag{5d}$$

The value X_2 is the private key which is calculated by the key leakage in (5d). It may be inconclusive which one of the transaction acts as tx_1 and tx_2. This though does not pose a problem, because with low effort two addresses can be calculated and listened to. In Fig. 2 the complete communication scheme is depicted.

One difference to the other methods is that here it would be useful if the bots have the sender address pre-configured, so that they only need to search for matching r_1 and r_2 values within the transactions of one source. In contrast to the methods above where on equal r values need to be detected, here it would be necessary to check all signature for r values that fulfill the relation $k_2 = r_1 \oplus X_1$ with $r_2 = k_2 \cdot G$. Checking for those conditions over all transactions is computational much more expensive than just checking for duplicate r values. This method does not need the secret sharing scheme for the same reasons it was introduced before. It can be used to separate the transactions with key leakage from the ones carrying the subliminal channel. In this case the bots see only two transactions with key leakage, what makes it faster and easier for the bots to determine the leaked key.

E. Cross-Blockchain Obfuscation

Since our method does not depend on a feature of a specific blockchain, it can even be distributed over multiple blockchains. Key leakage and message transmission could for instance be realized by utilizing multiple blockchains which use the same signature scheme. Blockchain parsers and tools for detecting conventional key leakage are mostly limited to one currency at a time. It would raise the effort for detection, but would not significantly increase the complexity of the communication scheme. The bots can be instructed to observe a set of blockchains in a certain order or all at once. Also, the blockchain could be changed from one botnet command to the other in order to exploit features of a specific chain such as low transaction cost or low block time (and therefore latency) or even to just further obfuscate the plain existence of the botnet.

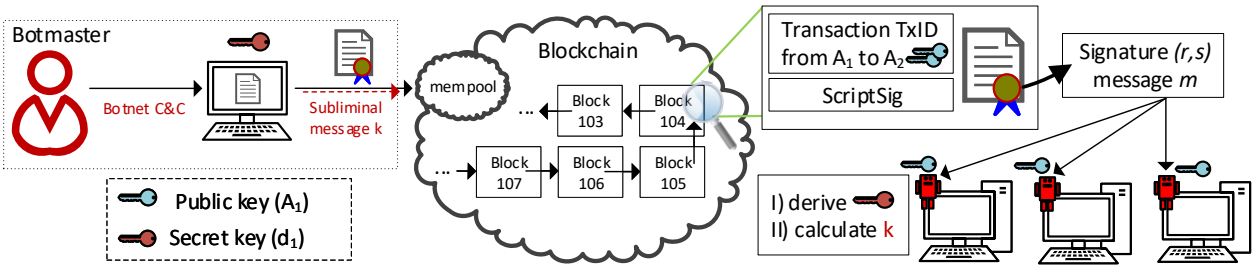

Fig. 2. Subliminal message propagation: the transaction is signed by botmaster master with a determined nonce k as subliminal message. The transaction can be seen in the mempool or as part of a mined transaction and is found by its address and processed by bots.

VI. Proof of Concept

In order to demonstrate the applicability of our method we crafted a transaction and injected a subliminal message in a signature that we pushed to the Bitcoin blockchain. We used the open source Bitcoin client *Electrum*[1] and modified its code to embed a subliminal message. As this is meant to be a proof of concept rather than a fully fledged implementation we implemented the show case of embedding a subliminal message with concealed key leakage in the Bitcoin blockchain. A full implementation should also prompt for the commands, handle the key usage, encoding and address usage to fit the C&C infrastructure and also handle various blockchains.
The receiver requests transactions with the provided address from which the key leakage is performed as source. This can be done by sending requests as a Bitcoin node or by requesting transactions from blockchain explorer websites.
The receiver then calculates the shared secret X_2 which is leaked. After that the shared secret X_1 and X_2 are combined to derive the private key and address where the subliminal message is hidden.

As depicted in Fig.3, the bots posses the pair $(X_{1,2}, A_1)$. For our proof we used the following values:
$A_1 =$ with A_1 being the address that leads to the transactions on which the key leakage takes place ($X_{1,2}$ denotes the X_1 part for recovering the private key for the address A_2. The content of this transaction from A_1 is $X_{2,2}$, which the bots combine with the pre-configured value $X_{1,2}$ to retrieve the private key of the address A_4 which is the source of the subliminal messages, i.e., from $X_{1,2}$ and $X_{2,2}$ the bot calculates private key d_4. With this private key the bot can also derive the public key and therefore the sender's address A_2 which is the source of the subliminal messages. In our case this leads the bots to the transaction described by the addres in table I. After this, transactions with A_2 as source address are fetched in which the subliminal messages are encoded. The signatures from the transactions from A_2 together with the leaked private key d_2 can then be used to calculate the

[1]Available under https://electrum.org

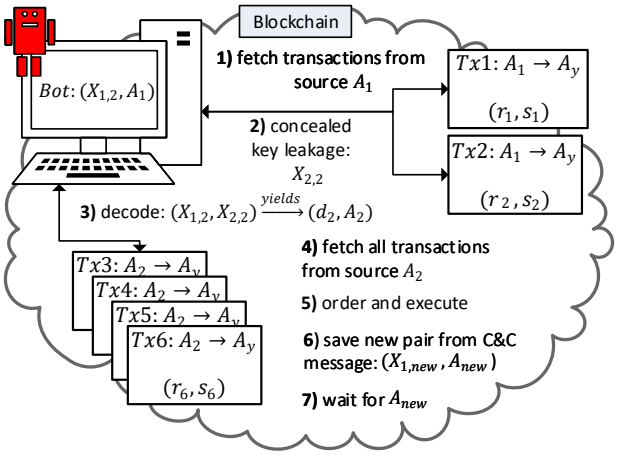

Fig. 3. Transaction and processing steps at a bot using simple secret sharing as used in our proof of concept. In this case the address A_1 from which the key is leaked is pre-configure at the bot (alternatively the bot could just search for duplicate r values in all transactions). The addresses A_y stand for arbitrary receiver adresses for the transactions. The subliminal information is injected by address A_2. $X_{1,2}$ stands for the first part (X_1) and $X_{2,2}$ for the second part (X_2) that is required to derive the private key for Address A_2. $X_{1,2}$ is pre-configured at the bot and $X_{2,2}$ is leaked by concealed key leakage. The subscripts i of (r_i, s_i) correspond to the i^{th} transactions.

C&C messages stored in the nonce k of the transaction with equation (2).

To make the example comprehensible, we encoded our message as 8-bit ASCII. With the information provided in table I the reader can simply fetch the transactions, calculate the shared secret, combine it with the values in table I and decode our subliminal message from the public Bitcoin blockchain.

VII. Discussion

We describe the usage of signatures in blockchains as C&C infrastructure for botnets. One important feature of the proposed communication scheme is, that it solely relies on digital signatures. It does not need fields of specific blockchain implementations such as the OP_RETURN field in Bitcoin and

TABLE I
VALUES FOR THE PROOF OF CONCEPT

Address concealed key leakage Tx1 & Tx2	A_1	1ArDRvW3f754DsC1Bawm4WnrsC1dyft8nJ
Address subl. message Tx3	A_4	1Ahg5AkeNJHorpfvzUmGRqNZgGtC9BGzdQ
Shared secret (bot)	$X_{1,4}$	0x5d8df0095ce7ec162ef205f6f69671e515af9fab3f90426e6ed9accf71a1651d
Shared secret (transmitted, key leakage)	$X_{2,4}$	0x7ca41f10674d555ffb807498a9a1c4760e1675ae0def9a0376408065299ddda5
Private key subl. message (calculated)	d	0x3e77c102528282cc62639755438b1f541d48c9a87130ead96772d939b9a4ec95
Signature key leakage Tx1	r_1	0xd2920fbb1dca14c9ece9ca83c4737a412b628065c077cf8bf5b92aa909c307cd
Signature key leakage Tx2	r_2	0x6ed704a3153ced68e8454151b4a4aedece1701b222387329704225de511d5b06
Signature key leakage Tx2	s_2	0x2232795ad015d573152566787193489e6e5aad99882bfef0f1b8a6cd60418dfe
Signature subl. channel Tx3	r_3	0x8b1ca9d68b18ce0609c2e5c68e666471fdd88545017cb37cc524ebf7e85b8f6f
Signature subl. channel Tx3	s_3	0x63cfb8d9ff34f0e3df358cf31bc0b05f8a7da47186041b604b511e36e7e2ef09
Hash of message to sign Tx2	z_2	0x44fd7052dca5ff1dba24a25e985e7ff4d233e7383acc7af58a9404d12642f9f6
Hash of message to sign Tx3	z_3	0x99cc41ba56754b7c04af25285a91f8b7f8da522398bc1ef92a0f4d63f7c5f538
Order of curve (secp256k1 [14])	n	0xfffffffffffffffffffffffffffffffebaaedce6af48a03bbfd25e8cd0364141

is therefore applicable to arbitrary blockchains and blockchain applications. Not only the content of the communication but also that fact that a communication took place between a sender and bots is hidden. Furthermore, the number of active bots can not be revealed by monitoring the C&C infrastructure since it usually is not recorded who is requesting which part of the blockchain.

Nevertheless, there are a few factors that influence the usability of our proposed method.

1) Transaction Fees: In blockchains for cryptocurrencies every transaction costs fees, which depend on the number and size of the transaction. So a botmaster would have additional costs for the botnet operation. Nevertheless, the development and maintenance of any conventional or neoteric C&C infrastructure also produce costs in effort and money [1]. Additionally they are exposed to the risk of take-down, which would render the effort useless. The proposed scheme scales well with any number of bots, and has good availability, due to the flexibility provided by multiple blockchains and the blockchains themselves being decentralized. Furthermore, to take down the botnet, the corresponding blockchains need to be taken down as well, which significantly reduces the botnet's take down risk. Also the applicability to different blockchains provides the botmaster the choice to exploit blockchain application with no or lower fees. We therefore think that transaction costs are only a small drawback.

2) Bot Takeover: An adversary who is able to take over a single bot is able to retrieve the current (X_1, A) pair. With this knowledge the adversary is able to get the leaked key and can listen to the C&C communication. Also,one may argue that an adversary in possession of the leaked key could forge own messages and send own commands to the bots. Such forged messages could for instance contain fake future addresses where no new messages can arrive. In order to prevent this we propose to first inject the commands in the blockchain and only after the commands have been injected to leak the key. This way an adversary cannot inject wrong commands in the blockchain, because the key is only revealed to the bots when the commands are already in the chain. That way the key leakage is solely used as a trigger, after all the other transactions (commands) are pushed.

A further way to make bots more robust against attacks is by providing a list of $(X_{1,n}, A_n)$ values. As long as concealed key leakage is used, the X_n values and old addresses are deleted and the used addresses do not relate to each other, the history of sent messages can not be deduced by an adversary who takes over the bot. Only the messages after the take-over could be monitored, by following the incoming messages by knowing (X_1, A).

3) Latencies: In [12] it was measured, that about 90% of the bots (which are on running hosts) respond in 10 seconds. Since the bots can see the transactions while they are in the mempool awaiting confirmation, they do not have to wait until they are mined. With the proposed scheme messages some preparation time has to be accounted for, since the transactions containing the C&C messages have to be confirmed first. An adversary could try to attack the communication if the key is leaked to early, by offering higher transaction fees than the botmaster. However after that and key leakage is performed, the bots can be triggered in the same time span.

4) Selection of Suitable Blockchains: Suitable cryptocurrencies for the proposed scheme can be determined by the following factors: current transaction fees, number of transactions per time, confirmation speed, signature scheme, anonymity and pseudonymity. But also other blockchains are applicable. The number of transactions can have an impact on the latency if the number is high. Also a very low value would be counterproductive, since then frequent C&C communication (and therefore an unusually high number of transactions) could be seen as an anomaly in the blockchain, exposing it as suspicious behavior.

5) Nonce Randomness: In our experiments we inject subliminal messages as plaintext (e.g., ASCII coded) in the nonce k of the signatures. Nevertheless, this reduces the overall number of possible k values and if extensively used produces bias in the distribution of k values. So the k values can no longer be considered as randomly selected. Since using a non-random nonce k can weaken the security of an ECDSA signature, we propose to encrypt the subliminal message

before it is injected in the signature. This way we maintain randomness with the k values and avoid any biased patterns. As encryption and decryption key the private key can be used that is anyway shared with the bots in the key leakage step.

VIII. Detection and Mitigation

The most effective way to take down the C&C infrastructure is to prohibit blockchain traffic or take the blockchain down, which is very unlikely. The first option might work for relatively small networks, such as corporate networks with strict firewall rules. However communication would still be able over various public blockchain explorer websites.

A. Signature Specific Mitigation

The paper [5] discusses several methods for the detection of subliminal channels. Small and repeating nonce values may pose a threat in terms of detection. This can be caused by sending the same deterministic commands and thus generating small and/or repeating nonce values k. Even if the transactions are not related, the same r values are produced, because r is only dependent on the nonce as shown in IV. This methods of detection can be rendered useless by embedding some random padding in the messages or by encrypting the messages before they are injected (as explained in VII). The other data transmitted, such as $(X_{1,new}, A_{new})$ are supposed to be random anyway, so they do not produce suspicious patterns, when embedded in the per-message secret k.

Approaches for subliminal-free variants of ECDSA and EdDSA are reviewed in [5]). One idea is to authorize a *warden* as introduced by Simmons [2], who is allowed to not only check the signatures, but also participates in the signing[7]. Having a trusted third party requires a lot of resources and also counters the decentralized and trustless concept of most blockchain applications and thus is not a viable solution for most applications. The same is true for *zero-knowledge proof* [6] and *pre-published nonce points* as shown in [5].

B. Cryptocurrency Specific Mitigation

Even if a subliminal-free signature scheme is introduced, this would result in a hard-fork for a cryptocurrency [22]. This means that the update can be introduced by the developers, but it still must be accepted by 50% plus one node to become standard. Eventually hard-forks can split up the cryptocurrency which results in two independent blockchains, which can have unwanted consequences for the economy linked to cryptocurrencies.

Another method to take down a botnet would be to link the used coins to the botmaster. Methods to de-anonymize holders of cryptocurrency are being developed constantly by the authorities to battle black markets. Techniques as presented in [23] could be utilized to link a botmaster to an exchange, where the authorities could request the identity of a user.

Nevertheless, the receiver of coins can be a random unsuspecting user, which could eventually send the coins to an exchange, thus driving the attention away from the botmasters.

On the other hand the botnet could be financed by coins which can not be traded on exchanges, since they stem from criminal and malicious activities. This would not only secure the botmaster, but could also be used to launder money, which would further minimize the cost of such a botnet.

IX. Conclusion

In this paper we introduce a method to establish a hidden C&C infrastructure to control a botnet using signatures in public blockchains. The method is easy to implement, scalable and only based on signatures. It is applicable to a wide variety of blockchains and can even be distributed over multiple blockchains to increase obfuscation. We show how botnet commands and the key to decrypt the commands can be injected in the blockchain and decrypted by the bots. As a proof of concept we injected a subliminal message and a key into the public Bitcoin blockchain. Since commands are injected before the key is leaked, the method prevents any adversary that succeeded to take over individual bots to inject valid commands himself. Since digital signatures are an essential part of blockchains, the findings are not limited by the application but only on the used signature schemes. We here showed how blockchains can be used for botnet control. Nevertheless, the presented method can be easily adjusted to any other use case where information should covertly be distributed to a set of receivers.

Acknowledgment

Special thanks goes to Gernot Vormayr for contributing his feedback and knowledge on botnets and botnet traffic. Thanks also goes to Maximilian Bachl and Martin Mosbeck who participation in discussion and proofreading.

References

[1] G. Vormayr, T. Zseby, and J. Fabini. "Botnet Communication Patterns". In: *IEEE Communications Surveys Tutorials* 19.4 (Fourthquarter 2017), pp. 2768–2796. DOI: 10.1109/COMST.2017.2749442.

[2] Gustavus J. Simmons. "The Prisoners' Problem and the Subliminal Channel". en. In: *Advances in Cryptology*. Springer, Boston, MA, 1984, pp. 51–67. ISBN: 978-1-4684-4732-3 978-1-4684-4730-9. DOI: 10.1007/978-1-4684-4730-9_5.

[3] Gustavus J. Simmons. "Subliminal Communication Is Easy Using the DSA". en. In: *Advances in Cryptology — EUROCRYPT '93*. Lecture Notes in Computer Science. Springer, Berlin, Heidelberg, May 1993, pp. 218–232. ISBN: 978-3-540-57600-6 978-3-540-48285-7. DOI: 10.1007/3-540-48285-7_18.

[4] Bruce Schneier. *Applied Cryptography: Protocols, Algorithms and Source Code in C 20th Anniversary Edition*. Englisch. Anniversary edition. Indianapolis, IN: John Wiley & Sons Inc, May 2015. ISBN: 978-1-119-09672-6.

[5] Alexander Hartl, Robert Annessi, and Tanja Zseby. "A Subliminal Channel in EdDSA: Information Leakage with High-Speed Signatures". In: *Proceedings of the 2017 International Workshop on Managing Insider Security Threats*. MIST '17. New York, NY, USA: ACM, 2017, pp. 67–78. ISBN: 978-1-4503-5177-5. DOI: 10.1145/3139923.3139925.

[6] Jens-Matthias Bohli, Maria Isabel Gonzalez Vasco, and Rainer Steinwandt. "A Subliminal-Free Variant of ECDSA". en. In: *Information Hiding*. Lecture Notes in Computer Science. Springer, Berlin, Heidelberg, July 2006, pp. 375–387. ISBN: 978-3-540-74123-7 978-3-540-74124-4. DOI: 10.1007/978-3-540-74124-4_25.

[7] Yinghui Zhang et al. "Provably Secure and Subliminal-Free Variant of Schnorr Signature". en. In: *Information and Communication Technology*. Lecture Notes in Computer Science. Springer, Berlin, Heidelberg, Mar. 2013, pp. 383–391. ISBN: 978-3-642-36817-2 978-3-642-36818-9. DOI: 10.1007/978-3-642-36818-9_42.

[8] Alexander Hartl, Robert Annessi, and Tanja Zseby. "Subliminal Channels in High-Speed Signatures". In: *Journal of Wireless Mobile Networks, Ubiquitous Computing, and Dependable Applications (JoWUA)* 9.1 (Mar. 2018), pp. 30–53.

[9] Arvind Narayanan et al. *Bitcoin and Cryptocurrency Technologies: A Comprehensive Introduction*. English. Princeton: Princeton University Press, July 2016. ISBN: 978-0-691-17169-2.

[10] Andrew Sward, Ivy Vecna, and Forrest Stonedahl. "Data Insertion in Bitcoin's Blockchain". en. In: *Ledger* 3.0 (Apr. 2018). ISSN: 2379-5980. DOI: 10.5195/ledger.2018.101.

[11] Roman Matzutt et al. "A Quantitative Analysis of the Impact of Arbitrary Blockchain Content on Bitcoin". en. In: *Proceedings of the 22nd International Conference on Financial Cryptography and Data Security 2018*. Springer, 2018, p. 18.

[12] Syed Taha Ali et al. "ZombieCoin 2.0: Managing next-Generation Botnets Using Bitcoin". en. In: *International Journal of Information Security* (June 2017), pp. 1–12. ISSN: 1615-5262, 1615-5270. DOI: 10.1007/s10207-017-0379-8.

[13] Syed Taha Ali et al. "ZombieCoin: Powering Next-Generation Botnets with Bitcoin". en. In: *Financial Cryptography and Data Security*. Lecture Notes in Computer Science. Springer, Berlin, Heidelberg, Jan. 2015, pp. 34–48. ISBN: 978-3-662-48050-2 978-3-662-48051-9. DOI: 10.1007/978-3-662-48051-9_3.

[14] Daniel R. L. Brown. "SEC 2: Recommended Elliptic Curve Domain Parameters (Version 2.0)". In: *Standards for Efficient Cryptography Group (SECG)* (Jan. 2010).

[15] *Bitcoins the Hard Way: Using the Raw Bitcoin Protocol*.

[16] Don Johnson, Alfred Menezes, and Scott Vanstone. "The Elliptic Curve Digital Signature Algorithm (ECDSA)". en. In: *International Journal of Information Security* 1.1 (Aug. 2001), pp. 36–63. ISSN: 1615-5262. DOI: 10.1007/s102070100002.

[17] Daniel J. Bernstein et al. "High-Speed High-Security Signatures". en. In: *Cryptographic Hardware and Embedded Systems – CHES 2011*. Lecture Notes in Computer Science. Springer, Berlin, Heidelberg, Sept. 2011, pp. 124–142. ISBN: 978-3-642-23950-2 978-3-642-23951-9. DOI: 10.1007/978-3-642-23951-9_9.

[18] *Things That Use Ed25519 [Online]*. https://ianix.com/pub/ed25519-deployment.html [Accessed: 10-Apr-2018].

[19] Adi Shamir. "How to Share a Secret". In: *Commun. ACM* 22.11 (Nov. 1979), pp. 612–613. ISSN: 0001-0782. DOI: 10.1145/359168.359176.

[20] Ernest F. Brickell. "Some Ideal Secret Sharing Schemes". en. In: *Advances in Cryptology — EUROCRYPT '89*. Lecture Notes in Computer Science. Springer, Berlin, Heidelberg, Apr. 1989, pp. 468–475. ISBN: 978-3-540-53433-4 978-3-540-46885-1. DOI: 10.1007/3-540-46885-4_45.

[21] Joppe W. Bos et al. "Elliptic Curve Cryptography in Practice". en. In: *Financial Cryptography and Data Security*. Lecture Notes in Computer Science. Springer, Berlin, Heidelberg, Mar. 2014, pp. 157–175. ISBN: 978-3-662-45471-8 978-3-662-45472-5. DOI: 10.1007/978-3-662-45472-5_11.

[22] M. Sato and S. Matsuo. "Long-Term Public Blockchain: Resilience against Compromise of Underlying Cryptography". In: *2017 26th International Conference on Computer Communication and Networks (ICCCN)*. July 2017, pp. 1–8. DOI: 10.1109/ICCCN.2017.8038516.

[23] M. Moeser, R. Boehme, and D. Breuker. "An Inquiry into Money Laundering Tools in the Bitcoin Ecosystem". In: *2013 APWG eCrime Researchers Summit*. Sept. 2013, pp. 1–14. DOI: 10.1109/eCRS.2013.6805780.