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 foundation for a wide range of malicious activities on the Internet. Sophisticated Command and Control (C&C) 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 ChainChannels, a method for hidden 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 the keying material required for extracting the subliminal information can be transmitted privately to the bots while being stored on a public blockchain. As proof of concept, we inject a subliminal message and a key in the Bitcoin blockchain and show how this information can be extracted from the transactions.**

Our method allows to establish a hidden C&C infrastructure over blockchains and send instructions to all bots without leaving 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 redesigning 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 (i.e., 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 on the Internet. The aim of botnet operators is to produce an economically cheap, logistically feasible, hidden, fast, and robust Command and Control (C&C) network, which is rather difficult to obstruct in its functions and ideally impossible to take down. 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 the botnet [2][3].

In this paper, we present ChainChannels, a novel way of privately multicasting 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 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 the keying material (needed to extract the subliminal information) can be distributed secretly to the bots such that storing the private key in advance in a bot can be avoided and take-over by an adversary that acquired information from a compromised bot is prevented. An adversary can follow the communication with a compromised bot but cannot 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. ChainChannels is not restricted to a specific blockchain and is robust against takeover. Even with knowledge of the subliminal message and the key, botnet commands cannot be forged and the size of botnet remains unknown.

Our contribution can be summarized as follows:

- We introduce a new scheme, ChainChannels, to establish a hidden C&C infrastructure for multicasting 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 distributed even over multiple blockchains.
- We show as an example how ChainChannels can be applied using the classical Elliptic Curve Digital Signature Algorithm (ECDSA) that is commonly used in blockchains. ChainChannels can also be applied when other signature schemes are used, including modern high-speed signatures such as Edwards-curve Digital Signature Algorithm (EdDSA) and signatures schemes based on Multi-variate Quadratic Quasigroups (MQQ),

- We present a method to secretly transmit keying material to the bots that enables them to extract 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 conduct an experiment as proof of concept and inject a subliminal message and a key into the Bitcoin blockchain. This message provides an example how commands can be transmitted covertly to all bots.
- We discuss several mitigation approaches. Nevertheless, all methods have severe drawbacks and are not applicable to current blockchains.

In this paper we show the covert transmission of messages for C&C communication in a botnet. Nevertheless, our approach provides a general method for hidden distribution channels over blockchains and therefore can be applied to other scenarios in which communication channels should remain hidden.

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 used in the context of botnets are:

- The *botmaster*, who controls the botnet and aims to stay stealthy.
- The *bots*, which reside on infected hosts and use their resources to perform tasks as commanded by the botmaster, such as stealing private information or performing distributed denial of service attacks (DDoS).
- The *C&C infrastructure* is used to send commands to the bots. It can be centralized, a peer to peer (P2P), or a hybrid structure and may use simple Internet relay chat (IRC) channels, HTTP(S), or more sophisticated neoteric protocols [1].
- The *adversaries* are authorities or malicious individuals who try to take down or take over the botnet for different reasons.

The C&C infrastructure is the key to the robustness of a botnet against takedown or takeover.

B. Subliminal Channels in Signature Schemes

Simmons was the first to show how narrowband and broadband channels could be implemented in digital signatures [4, 5]. Simmons introduced a model with two communicating parties, so-called prisoners, who are monitored by a warden. The prisoners are allowed to communicate in clear text only but may use signatures to ensure integrity of the messages. The messages and the signatures are inspected by the warden. Simmons shows that a subliminal message can be embedded in a valid signature in such a way that the existence of the hidden message can not be detected by the warden.

In contrast to classic steganography, in which objects (such as images) are used to clandestinely transmit information, subliminal channels are embedded in cryptographic information and the resulting communication can not be distinguished from other not carrying the channel [6].

There exist narrowband and broadband subliminal channels. An example for a narrowband channel is to just try out different nonces until a predefined number of bits in the signature coincide with the subliminal information [7]. This works without sharing a secret key with the receiver but the computational effort rises exponentially with the number of bits, which makes this channel viable for very few bits only. In contrast, broadband channels may use the entire nonce to hide information. The main disadvantage is that usually the private key is needed by the receiver to extract the subliminal information. Since the private key is used to sign the transactions in the blockchain, sharing the private key enables the receiver of the subliminal information to forge own transactions. To prevent this, we will introduce a method to distribute the key without a receiver being able to forge own transactions beforehand. The two main digital signature schemes used for blockchain applications are ECDSA and EdDSA. For both it has been shown that they contain broadband subliminal channels [7, 8]. Several other signature schemes that could be used in blockchain applications may allow the injection of subliminal information as well [9, 10].

C. Data Insertion in Blockchains

Blockchains were initially used for cryptocurrencies. More applications developed on top of them, such as time stamping and notary services, but also malicious and illicit content emerged [11]. The Bitcoin blockchain was extensively analyzed for the insertion of arbitrary data [12, 13]. It was shown that the OP_RETURN field and payments to fake public key, key hashes, or script hashes 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 blockchains. Nevertheless, the content is transmitted overtly and can be observed by anyone.

Some previous work [14][15] 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. 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 offers low latency, is distributed and decentralized by design, has a consistent network state, and that it would be hard to censor C&C instructions without significantly impacting the overall function of the blockchain. Also, the proposed communication channel scales well with the number of bots and offers the botmaster an infrastructure that is not likely to be taken down and thus less risky and less costly.

Zombiecoin [15] is specific to the Bitcoin blockchain only. It uses the OP_RETURN field and proposes the use of a narrowband subliminal channel only, 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 entire nonce can be used to transmit subliminal information. Our method does not require any specific blockchain field as it relies solely on one thing that all blockchains have in common – digital 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. Nevertheless, ChainChannels is equally applicable to any other blockchain as long as it uses a suitable signature scheme (such as ECDSA) that allows the injection of subliminal messages. We make the following assumptions to establish the C&C infrastructure, which we find to be valid and to not reduce the generality of our results:

1) We focus our contribution on 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 as out of scope, i.e., the scanning for vulnerable hosts as potential new bots and the compromisation of hosts and initial propagation of bot software to compromised hosts can be done by a wide range of mechanisms 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(S), which can be implemented to reduce communication overhead or as a fallback option when bots do not have access to certain ports (e.g., TCP port 8333 for Bitcoin) due to firewall restrictions for example.
3) All bots are provided with at least one initial address on which they check for new commands. This initial address can be provided with the installation of the bot software. After a command is finished, the bots are provided with new information on where to look for subsequent commands. Newly infected bots need to have the current address, since they would follow through all commands starting from the initial transaction. Alternatively, newly infected bots and bots which were offline for some time can parse through the chain of messages, discard old commands, and only store the last one they can see.
4) All transactions are designed in a way, that no coins are left to claim after a private key has been exposed deliberately. 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 Sections IV and V.

IV. Subliminal Channels in Blockchain Signatures

As an example we use the public blockchain for the Bitcoin cryptocurrency. Like many other blockchains, the Bitcoin blockchain uses the ECDSA signature scheme with curve secp256k1 [16]. Since there are many variations and developments stemming from Bitcoin, the introduced subliminal channel is not limited to the original Bitcoin blockchain and independent 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 [17]:

1) A cryptographically secure nonce k is generated per message with $1 \leq k \leq n-1$. Measures have to be taken so that k can not be guessed by an adversary, which are left to the implementer.
2) A new point on the curve is computed by $(x_1, y_1) = k \cdot G$. 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 [18]. 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 of the nonce $k \mod n$ as follows:

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

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 nonce k, which is supposed to be random, is substituted 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 a random number. In order to extract the subliminal information from the signature the receiver needs to know the private key d. With knowledge of the private key, the subliminal information k_{msg} then can be extracted by the receiver as follows:

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

While the signature is valid and independent of the specific value chosen for k, it must be avoided to use repeating values of k, especially for the same private key. Repeating values of k may occur if the same unencrypted command and control messages are transmitted repeatedly. If the same k is used multiple times, the private key is leaked unintentionally [17].

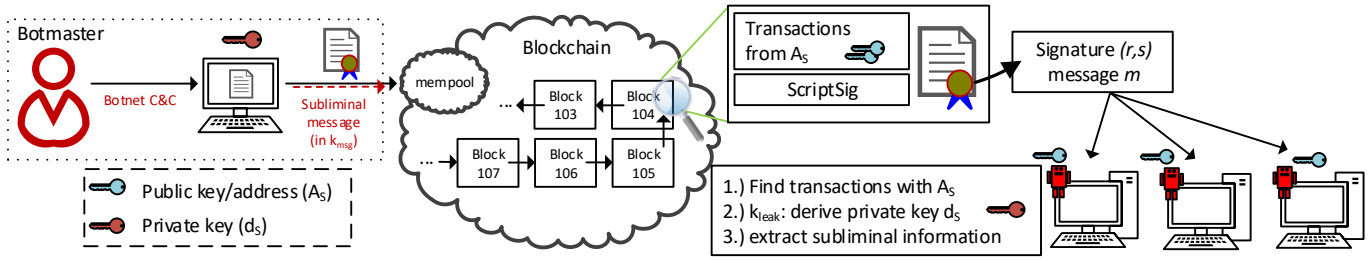

Fig. 1. **Subliminal message propagation:** The transaction is signed by the botmaster with a determined nonce k_{msg} as subliminal message. The transaction can be seen in the mempool or as part of a mined block and is found by its address and processed by bots. In the final transaction, the botmaster uses a nonce k_{leak} that is pre-configured in the bots. The bots can therefore derive the private key and extract the subliminal information encoded in the previous transactions.

In Section V, we show how the sender can leak the information about the private key d 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 only after it is safe to assume that the embedded subliminal message can not be altered anymore.

The further discussion and notation is referring 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, such as emerging high-speed signature schemes like EdDSA [19]. Especially the Ed25519 variation is used or planned to be used in various blockchain applications [20]. As demonstrated in [7], a broadband subliminal channel in EdDSA can be established similar to the one in ECDSA. MQQ-based signature schemes also allow the establishment of subliminal channels as demonstrated in [10]. Especially all the MQQ schemes that are currently considered secure are vulnerable to subliminal channels. Therefore, the basic idea and all following concepts can equally be applied to blockchains using EdDSA or MQQ-based signature schemes. The same goes for Schnorr signatures [9].

V. Distribution of the Private Key

Subliminal messages are pushed by the botmaster to the blockchain (or are awaiting confirmation in the mempool) and the bot knows from which address to expect the subliminal instructions. The propagation of such a message is depicted in Fig. 1. Nevertheless, as shown in Section IV the bots require knowledge of the private key d of the sender used to sign messages in order to extract the subliminal message. In this section, we present different methods how the private key can be distributed to the bots.

A. Pre-Configuration of the Private Key

The easiest method is 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 are valid and indistinguishable from that by the actual botmaster.

B. Key Leakage Based on Nonce Reuse

A better method to provide the key information to the bots is to send the private key after the botnet is established. In this way, the key needs to be revealed only shortly before the commands should be executed. For this, the bots first just collect all messages received from the botmaster without extracting the subliminal information. Then, as soon as they receive the private key, they extract the subliminal messages before triggering the execution of the commands. In his way, the private key remains unknown to the bots and undetectable for any adversary until botnet actions are triggered.

A method to distribute the private key to the bots is to use a classical key leakage method based on nonce reuse, which is conducted 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) for two distinct messages m_1 and m_2 but with identical values of r. The calculation of the private key from such two messages is presented in [17]. The receiving bots are looking for all transactions from the sender address and check the r values. When the 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 may also generate identical k values by accident, which 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. For this reason, one drawback of the key leakage method based on nonce reuse is that other blockchain users may notice the two identical r values as well by parsing the blockchain and may therefore derive the leaked private key. Subsequently, also the messages transmitted to the bots

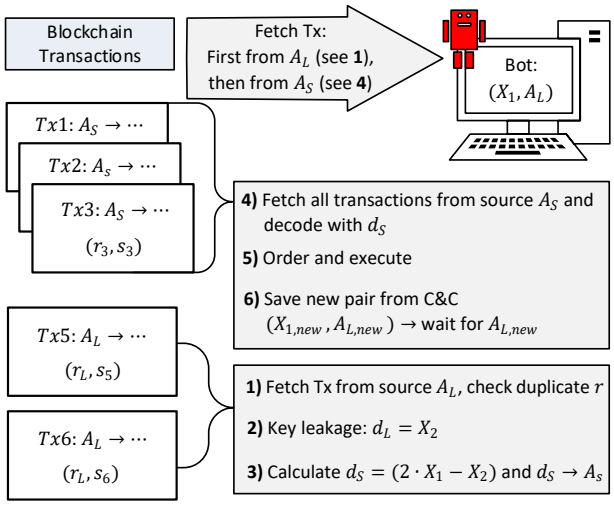

Fig. 2. **Key Leakage with Secret Sharing:** Transaction and processing steps at a bot using key leakage with simple secret sharing. Tx stands for transaction. Here only the transactions from the address that injects the subliminal information (A_S) and from the address (A_L) that leaks the key are shown. The transactions from (A_S) have been inserted in the blockchain before the key is leaked. The transaction receivers are arbitrary and can vary. The bot first fetches transactions from the leaking address A_L, and check for duplicate r that leak the key (d_L) of A_L. It uses the leaked X_2 (which corresponds to d_L) to calculate the secret key d_S. From d_S it also can derive the address of the subliminal sender A_S. With d_S the bot can then decode the subliminal messages from transactions from A_S.

would be visible (and decodable) to anyone with access to the blockchain who knows or suspects that a subliminal channel is used. For this reason, 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 subsequent subsections.

C. Key Leakage Based on Secret Sharing

To solve the problem of exposing the C&C messages to the public, a secret sharing scheme can be used. There exist various secret sharing schemes such as Shamir's [21] or Brickell's [22]. For the problem at hand any secret sharing scheme can be employed without affecting the basic principle of operation. For a simple secret sharing scheme the 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, which defines the upper limit of d and R as seen in Eq. (3a). Since the size of the key to be split is 32 byte, the (prime) order n of the secp256k1 curve [16] can be used. The private key d can easily be reconstructed when both parts, X_1 and X_2, are known (see Eq. (3d)).

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

We now assume that all bots know the initial address A_L from which the key will be leaked and also know the value X_1 initially, which can be distributed to the bots in the same way as the initial address A_L (see Section III). The important difference to the key leakage based on nonce reuse is that only bots know the common X_1 and the information leaked (X_2) is useless without the common secret X_1. In this way, a part of the secret can simply be pre-configured in the bots code without the danger that an adversary gets knowledge of the entire private key as soon as a bot gets compromised. Nevertheless, the first part of the secret (X_1) is required to calculate the private key while only the second part of the secret (X_2) is leaked through nonce reuse.

One important detail with all secret sharing schemes is that the address from which 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_L anyone can derive the private key d_L for A_L, which means that an adversary could wait for the key leakage, derive the private key, and then inject subliminal messages to take over the botnet. If the subliminal messages are sent from a different address (e.g., A_S), however, then X_1 is required together with the X_2 to calculate the private key d_S of A_S (Eq. (3d)), and only bots that had the X_1 pre-configured can decode the subliminal messages. The processing steps for key leakage based on secret sharing are shown in Fig. 2.

D. Concealed Key Leakage With a Pre-Shared Nonce

Until now, we have secured the subliminal messages from being directly exposed to the public while not requiring the (entire) private key to be pre-configured in the bots. One problem remains though: the existence of nonce reuse alone may raise suspicion, especially given the fact that blockchains are constantly monitored for nonce reuses, mainly with the aim to steal any coins associated with a leaked key [23]. While not dangerous to the operation of the ChainChannels C&C infrastructure, nonce reuse may draw attention to the botnet, which is unwanted. To stay covert and to not produce suspicious transactions, we propose to use an advanced method to avoid using the same nonce k twice.

For this purpose, we intentionally violate one assumption regarding ECDSA – that the nonce must be unpredictable. What we suggest for ChainChannels is to pre-configure a random secret nonce k_{leak} at the bots (in addition to the address). It is known only at the bots and the botmaster and remains unpredictable to everyone else. In addition, we now do not need to use different addresses but can do the key leakage from the same address A_S from which we sent the subliminal messages.

The sender then is using transactions from address A_S to send commands and to leak the key. It first encodes all commands in nonces k_{msg} and generates the matching signatures (r, s) as shown in Eq. 1. Only after all commands have been inserted in the blockchain (and have been confirmed) the botmaster uses the pre-shared k_{leak} to generate the final

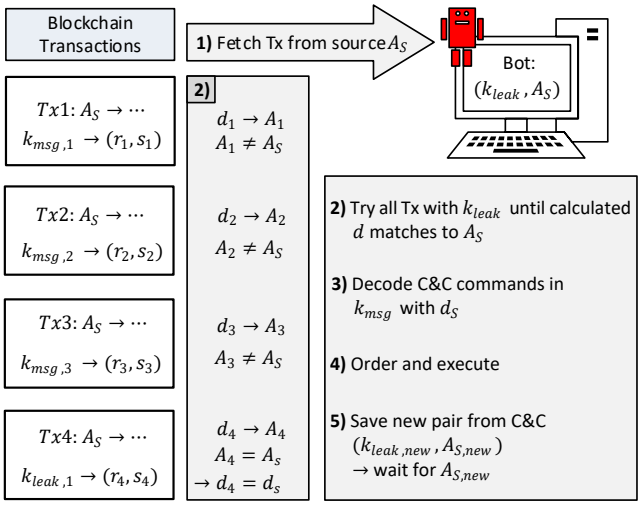

Fig. 3. **Concealed Key Leakage:** Transaction and processing steps at a bot using concealed key leakage (as later used in our proof of concept). Only transactions from A_S are shown. First the subliminal messages are injected by address A_S and later a signature is generated with k_{leak} to leak the secret key d_S of A_S. The bot continuously fetches transactions from A_S and calculates a potential secret key d from the signatures (r, s) and nonce k_{leak}. If the signature have been generated with k_{leak} the calculation of d will lead to the correct secret key d_S of A_S. The key d_S can then be used to decode the commands in the previous transactions.

signature (r_{leak}, s_{leak}) and with this leaks the private key needed to decode the previous commands.

The bot knows the initial address A_S and the pre-configured nonce k_{leak}. It checks all transactions from A_S and extracts the signatures. For each signature (r, s) it calculates a potential secret key d by using the pre-shared secret nonce k_{leak} as shown in Eq. 4.

$$d = r^{-1} \cdot (s \cdot k_{leak} - z) \mod n \tag{4}$$

The bot then checks if the the address A_S can be derived from the calculated d. This is done by first calculating the matching public key from the secret key d. Then from the public key the address can be derived by the blockchain rules [24]. The address that was derived from d is then compared to the pre-configured address A_S. If the signature was created with a $k_{msg} \neq k_{leak}$ the resulting d is not the correct secret key and the resulting address will be different from A_S. So those signatures contain the actual commands and are not (yet) the leakage of the private key. The signatures are just stored by the bot for later decoding.

Only for the last transaction (as soon as the sender wants to leak the key) the sender used k_{leak} as nonce to calculate the signature. So when the bot now applies Eq. 4 it will actually get the correct secret key d_S for address A_S. The bot recognizes, that it is the correct key because if it derives the address from the key, it will get the correct address A_S. The bot can now use the secret key d_S to decode all messages that had been encoded in the signatures of the previous transactions. The processing steps for the concealed key leakage are shown in Fig. 3.

To an observer the signatures of the ChainChannels C&C infrastructure are completely indistinguishable from signatures of regular transactions. They neither trigger any key leakage detection nor raise any suspicion. The bots, however, who know the address and the pre-shared nonce can derive the private key and therefore extract the subliminal information included in the signatures of previous transactions. In order to keep ChainChannels functional for multiple botnet commands, the subliminal information in the messages from the initial address should include a new address to listen for transactions as well as a new pre-shared nonce. In this way, ChainChannels can be used indefinitely.

E. Cross-Blockchain Usage

Since ChainChannels does not depend on features of a particular blockchain, it can even be distributed over multiple blockchains. Key leakage and message transmission could for instance be realized by utilizing multiple blockchains (as long as the blockchains employ signature schemes where subliminal channels can be exploited). Blockchain parsers and tools for detecting conventional key leakage are mostly limited to one blockchain at a time. So using multiple blockchains would further 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.

VI. Proof of Concept

In order to demonstrate the applicability of ChainChannels 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-fledged 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 receives transactions with the provided address as source. This can be done by sending requests as a Bitcoin node or by requesting transactions from blockchain explorer websites. As depicted in Fig.3, the bots possess the pair (k_{leak}, A_S). The receiver tries to calculate the private key d with the known k_{leak} for every transaction from A_S (see Eq. 4).

For our proof of concept we used the values listed in Table I, with A_S being the address that leads to the transactions with the subliminal message and concealed key leakage and k_{leak} denotes the pre-shared nonce for address A_S. Other values

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

such as the signature pair (r_i, s_i) or the transaction hash z_i can be derived from the transactions. All transactions before the final transaction include subliminal information that is stored (and extracted later). In the final transaction from A_S the concealed key leakage takes place. For this reason, we can derive the private key d_S and extract the C&C messages encoded in the nonces of the previous transactions (Eq. 4). 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, read out other needed values, calculate the leaked key with the known nonce, and decode our subliminal message from the public Bitcoin blockchain.

VII. Discussion

We described 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 a particular blockchain implementations such as the OP_RETURN field in Bitcoin and is therefore applicable to arbitrary blockchains and blockchain applications. Not only the content of the communication is hidden but also that fact that communication took place at all between the botmaster and the bots. Furthermore, the number of active bots can not be revealed by monitoring the C&C infrastructure since it is usually not recorded who is receiving which part of the blockchain. Nevertheless, there are a few factors that influence the usability of ChainChannels, which we will discuss next.

1) Transaction Fees: In blockchains for cryptocurrencies every transaction costs some fees, which depend on the size of the transaction (and on the particular blockchain). For this reason, 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, the botnets are exposed to the risk of take-down, which would render the effort useless. ChainChannels scales perfectly with the number of bots and has excellent 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 applications with low 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 $(k_{leak,i}, A_i)$ pair. With this knowledge, the adversary is able to get the leaked key and can listen to the C&C communication. Also, one could argue that an adversary in possession of the leaked key may be able to 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 to leak the key only after the corresponding blocks have been confirmed. That way the key leakage is solely used as a trigger after all the other transactions (commands) are pushed. In this way, an adversary can inject own commands into the blockchain only after because the key has been revealed to the bots, which means that the bots are already listening for transaction with the new source address.

Another way to make bots more robust against attacks is providing a list of $(k_{leak,i}, A_i)$ values. As long as concealed key leakage is used, the $k_{leak,i}$ 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.

3) Transaction Malleability: Parts of the ECDSA signatures in Bitcoin and similar blockchains can be changed without voiding its validity. Furthermore, numerous attacks occurred where this was abused. Transactions were intercepted, modified and transmitted by the attacker to pass the same transaction with another transaction ID [25].
Bitcoin therefore enforces low values of s to combat signature malleability. If a value s is larger than half the curve order n, the value $s_{new} = n - s$ is used instead [26]. The signature is still valid, but the subliminal channel cannot be calculated as described in Eq. 2. When decoding a subliminal message, the s value has to be checked for the mentioned condition, and changed back as needed to ensure reliable decoding of messages.

4) Latencies: In [14] 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 ChainChannels scheme some preparation time has to be accounted for since the transactions containing the C&C messages have to be confirmed first before leaking the key. An adversary could try to attack the communication if the key is leaked too early, by offering higher transaction fees than the botmaster. Nevertheless, after the key leakage is performed, the bots can be triggered in the same time span.

5) Selection of Suitable Blockchains: Suitable blockchains for ChainChannels can be determined by the following factors: current transaction fees, number of transactions per time, confirmation speed (i.e., block time), signature scheme, and anonymity. 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.

6) Nonce Randomness: In our experiments we inject subliminal messages as (ASCII encoded) plaintext in the nonce k of the signatures. Nevertheless, this reduces the overall range of possible k values and if extensively used produces bias in the distribution of k values such that the k values can no longer be considered randomly selected. Since using a non-random nonce k can weaken the security of an ECDSA signature, we

TABLE I
VALUES FOR THE PROOF OF CONCEPT

Source Address (message & leakage)	A_S	1Ahg5AkeNJHorpfvzUmGRqNZgGtC9BGzdQ
Pre-shared secret nonce for leakage	k_{leak}	0x51b094cca21e39f76f50486841a315284669ff0f2b3481b9720e8d18443a2ee7
Private key subl. message (calculated)	d_S	0x3e77c102528282cc62639755438b1f541d48c9a87130ead96772d939b9a4ec95
Order of curve (secp256k1 [16])	n	0xfffffffffffffffffffffffffffffffebaaedce6af48a03bbfd25e8cd0364141
TxID subl. message (block 508748)	$TxID_1$	0xbfdd501624f574662f67a57aa0b1db48bfa82f8a2c949ed8978970c49a76cf92
TxID key leakage (block 527646)	$TxID_2$	0x41945988509707921278fbff9254d468e634062b82fa5459c3e8bf12909a1955

propose to encrypt the subliminal message before it is injected in the signature. In 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 ChainChannels is to prohibit blockchain traffic in general or take the particular blockchain down, which is very unlikely. The first option might work for relatively small networks with strict firewall rules, such as corporate networks. However, communication would still be able over public blockchain explorer websites.

A. Signature Specific Mitigation

In [7] several methods for the detection of subliminal channels are discussed. 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. Even if the transactions are not related, the same r values are produced, because r is only dependent on the nonce as shown in Section IV. This method 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 Section VII). The other transmitted data, such as $(X_{1,new}, A_{new})$, are supposed to be random anyway so that they do not produce suspicious patterns when embedded in the per-message nonce k.

Approaches for subliminal-free variants of ECDSA and EdDSA are reviewed in [7]. One idea is to authorize a *warden*, as introduced by Simmons [4], who is allowed to not only check the signatures but also participates in the signing [9]. 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 proofs* [8] and *pre-published nonce points* as shown in [7].

B. Blockchain Specific Mitigation

Even if a subliminal-free signature scheme is introduced, this would result in a hard-fork for a blockchain [27]. This means that the update can be introduced by the developers but still must be accepted by 50% plus one node to become standard. Eventually, hard-forks can split up the blockchain, which results in two independent blockchains and can have unwanted economic consequences. 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 criminals. Techniques as presented in [28] could be utilized to link a botmaster to an exchange, where the authorities could request the identity of a user. Nevertheless, since the destination address in transactions can be arbitrary the receiver of coins can be a random unsuspecting user, who could eventually send the coins to an exchange, thus driving the attention away from the botmaster. 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 introduced the ChainChannels scheme to establish a hidden C&C infrastructure to control a botnet using signatures in public blockchains. ChainChannels is easy to implement, scalable, and only based on digital 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 private key to extract 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. Since digital signatures are an essential part of blockchains, the findings are not limited by the application but only by 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 for hidden multicast communication, where information should covertly be distributed to a set of receivers.

Acknowledgment

We would like to thank Gernot Vormayr for contributing his feedback and knowledge on botnets and botnet traffic as well as Maximilian Bachl and Martin Mosbeck who participated in discussions 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.

[2] C. Czosseck, G. Klein, and F. Leder. “On the Arms Race around Botnets - Setting up and Taking down Botnets”. In: *2011 3rd International Conference on Cyber Conflict*. June 2011, pp. 1–14.

[3] Y. Nadji, R. Perdisci, and M. Antonakakis. "Still Beheading Hydras: Botnet Takedowns Then and Now". In: *IEEE Transactions on Dependable and Secure Computing* 14.5 (Sept. 2017), pp. 535–549.

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

[5] Gustavus J. Simmons. "Subliminal Communication Is Easy Using the DSA". 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.

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

[7] 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. ACM, 2017, pp. 67–78. ISBN: 978-1-4503-5177-5.

[8] Jens-Matthias Bohli, Maria Isabel Gonzalez Vasco, and Rainer Steinwandt. "A Subliminal-Free Variant of ECDSA". 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.

[9] Yinghui Zhang et al. "Provably Secure and Subliminal-Free Variant of Schnorr Signature". 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.

[10] 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.

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

[12] Andrew Sward, Ivy Vecna, and Forrest Stonedahl. "Data Insertion in Bitcoin's Blockchain". In: *Ledger* 3.0 (Apr. 2018).

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

[14] Syed Taha Ali et al. "ZombieCoin 2.0: Managing next-Generation Botnets Using Bitcoin". In: *International Journal of Information Security* (June 2017), pp. 1–12.

[15] Syed Taha Ali et al. "ZombieCoin: Powering Next-Generation Botnets with Bitcoin". 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.

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

[17] Don Johnson, Alfred Menezes, and Scott Vanstone. "The Elliptic Curve Digital Signature Algorithm (ECDSA)". In: *International Journal of Information Security* 1.1 (Aug. 2001), pp. 36–63.

[18] *Bitcoins the Hard Way: Using the Raw Bitcoin Protocol [Online]*. http://www.righto.com/2014/02/bitcoins-hard-way-using-raw-bitcoin.html [Accessed: 07-Jun-2018].

[19] Daniel J. Bernstein et al. "High-Speed High-Security Signatures". 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.

[20] *Things That Use Ed25519 [Online]*. https://ianix.com/pub/ed25519-deployment.html [Accessed: 07-Jun-2018].

[21] Adi Shamir. "How to Share a Secret". In: *Commun. ACM* 22.11 (Nov. 1979), pp. 612–613.

[22] Ernest F. Brickell. "Some Ideal Secret Sharing Schemes". 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.

[23] Joppe W. Bos et al. "Elliptic Curve Cryptography in Practice". 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.

[24] F. Tschorsch and B. Scheuermann. "Bitcoin and Beyond: A Technical Survey on Decentralized Digital Currencies". In: *IEEE Communications Surveys Tutorials* 18.3 (thirdquarter 2016), pp. 2084–2123.

[25] Christian Decker and Roger Wattenhofer. "Bitcoin Transaction Malleability and MtGox". In: *Computer Security - ESORICS 2014*. Lecture Notes in Computer Science. Springer, Cham, Sept. 2014, pp. 313–326. ISBN: 978-3-319-11211-4 978-3-319-11212-1.

[26] Pieter Wuille. *BIP 0062: Dealing with Malleability - (2014) [Online]*. https://github.com/bitcoin/bips [Accessed: 10-Jun-2018].

[27] 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.

[28] 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.