Utilizing Public Blockchains for Censorship-Circumvention and IoT Communication

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

Bingsheng Zhang
Lancaster University, UK
b.zhang2@lancaster.ac.uk

***Abstract*—The advancement of blockchain technology and the Internet of Things (IoT) has presented us with unlimited possibilities to integrate the physical and the virtual worlds. In this work, we demonstrate how to override *any* existing public blockchain, that has enough redundancy in its transactions, to covertly broadcast arbitrary information. Besides, we implement and demonstrate our technique on a real-world cryptocurrency and show how it can be utilized in two specific applications: recording IoT data, and circumventing censorship.**

***Index Terms*—blockchains, IoT, censorship circumvention, broadcast communication.**

I. Introduction

Blockchains have presented a major paradigm shift in realizing distributed and append-only immutable ledgers. And ever since the inception of Bitcoin [25], blockchains have not been used in cryptocurrencies only, but also in smart contracts [38], medical records [18], identity verification [6], the insurance sector [13], and other fields. On the other hand, the Internet of Things (IoT) has been perpetually bridging the gap between the physical world and the virtual or digital world by interconnecting and automating all objects around us, from smart homes [19] to smart cities [32]. However, there remain some technical challenges that prevent the full realization of the potential of IoT. Examples of these challenges include synchronization among IoT devices, access control, security, and durability.

In this work, we present how *any existing* public blockchain application, with enough bandwidth in its transactions, can be overridden and utilized in many scenarios. We specifically demonstrate the application of Skywhisper in two use cases: 1) recording IoT data, and 2) circumventing censorship. Also, we evaluate the security and performance of our construction. We call our system *Skywhisper*, and to the best of our knowledge, it is the world's first covert broadcast channel that is unobservable to any third-party auditor.

System overview. The main idea behind Skywhisper is exploiting public blockchains and using them as a carrier for arbitrary information. Indeed, with the benefit and promising future of blockchain technology, most countries, even China, cannot afford to completely stop the adoption and usage of blockchains. Furthermore, the absence of a central authority in decentralized blockchains makes it hard to censor the posted content. Nevertheless, merely encrypting the message and attaching the ciphertext as a payload to the transactions is detectable. Hence, to resolve this issue, we have observed that the uncontrolled randomness used in blockchains' cryptographic primitives, such as the signatures attached to the transactions, can be exploited to hide arbitrary messages. Equally important, this approach is provably secure in the sense that a transaction embedded with a hidden message is indistinguishable from a normal transaction to a computationally bounded auditor.

Since our technique is to modify the cryptographic components in the blockchain transactions, the throughput largely depends on the size of these cryptographic components. After a systematic study, we have identified that CryptoNote-based blockchains [31] are ideal candidates. The CryptoNote framework uses ring signatures to enhance privacy, and the random group elements inside the ring signatures can be easily manipulated to embed arbitrary messages. Therefore, the Skywhisper prototype is developed on Bytecoin [3] which is based on the CryptoNote protocol. In addition, to enable the broadcast feature, we adopted the state-of-the-art collusion-resistant broadcast encryption scheme by Boneh et al. [2]. Notably, Skywhisper does not hinder standard Bytecoin functionalities; hence, anyone can use Skywhisper for covert broadcast communication and/or as a cheap hidden persistent storage along with their normal Bytecoin transactions. For instance, Skywhisper can be used for uncensorable cyberlockers. At the time of submission, persistently storing 1MB of data on Bytecoin and using its P2P network as CDN costs about $\$0.05$. Furthermore, Skywhisper can establish an unbounded number of channels, and each channel can support hundreds of subscribers. This enables Skywhisper to be used in various scenarios.

II. Preliminaries

Blockchain. The term *blockchain* encompasses a broad range of distributed ledger technologies initiated by Bitcoin [25]. There are two types of blockchains; permissioned (private) and permissionless (public). In this work, we mainly focus on permissionless blockchains. Typically, a permissionless blockchain uses a *Proof-of-X* mechanism, such as Proof-of-Work (PoW) and Proof-of-Stake (PoS), to randomly nominate a node which will propose the next block. The valid transactions contained in a block need to be signed by the owner(s) of the corresponding consumed coins. Most blockchain systems use randomized signature algorithms, which makes our technique applicable to a wide range of blockchain applications.

```
function Sign($\{P_i\}_{i=1}^{k}, t_s, s, m$):
  • Set $I := \mathsf{hash}_g(P_s)$;
  • For $i \in [k]$, pick $q_i \overset{\$}{\leftarrow} \mathbb{Z}_p$;
  • For $i \in [k], i \neq s$,pick $w_i \overset{\$}{\leftarrow} \mathbb{Z}_p$;
  • For $i \in [k]$:
      – Set $L_i := g^{q_i}$ if $i = s$;
        Set $L_i := g^{q_i} \cdot P_i^{w_i}$ if $i \neq s$;
      – Set $R_i := (\mathsf{hash}_g(P_i))^{q_i}$ if $i = s$;
        Set $R_i := (\mathsf{hash}_g(P_i))^{q_i} \cdot I^{w_i}$ if $i \neq s$;
  • Set $c := \mathsf{hash}_p(m, L_1, \ldots, L_k, R_1, \ldots, R_k)$;
  • For $i \in [k]$:
      – Set $c_i := w_i$ if $i \neq s$;
        Set $c_i := c - \sum_{i=1}^{k} c_i$ if $i = s$;
      – Set $r_i := q_i$ if $i \neq s$;
        Set $r_i := q_s - c_s t_s$ if $i = s$;
  • Return $\sigma := (I, c_1, \ldots, c_k, r_1, \ldots, r_k)$.
end function
```

Fig. 1: CryptoNote Signing Algorithm. $\{P_i\}_{i=1}^{k}$ are public keys, t_s is the signer's private key, and s is his index

IoT. The concept of the 'Internet of Things' (IoT) was coined in 1999 by a researcher who was working on radio-frequency ID (RFID) devices and their interconnection with the internet [11][1]. In this work, whenever we use the term IoT, we refer to any object in the physical world that may be connected to the internet and integrated with the virtual world.

Brief description of CryptoNote. CryptoNote is a protocol proposed by Nicolas van Saberhagen [31], and it has been implemented in many emerging cryptocurrencies, such as Bytecoin [3], CryptoNoteCoin [8], etc.

As depicted in Fig. 1, for a ring of size k, the format of the CryptoNote ring signature is $\sigma = (I, c_1, \ldots, c_k, r_1, \ldots, r_k)$. Suppose the sender's public key is P_s, $s \in [k]$, then for all $i \in [k]$ and $i \neq s$, the components c_i and r_i are uncontrolled random group elements in $\mathbb{Z}_p$. *These random elements are highlighted in gray, and they are most important to our work, as they are used to convey arbitrary information.* For more details on CryptoNote's ring signature, please refer to CryptoNote's white-paper [31].

Steganography. Steganography refers to the techniques that allow a sender to send a message covertly over a *communication channel* so that the mere presence of the hidden message is not detectable by an adversary who monitors the channel [15][9]. Modern steganography techniques can be applied to various media, such as images, audios, HTML files, etc. A stegosystem consists of three PPT algorithms $\mathcal{ST} := (\mathsf{KeyGen}, \mathsf{Embed}, \mathsf{Extract})$ as follows:

- $(\mathsf{ek}, \mathsf{dk}) \leftarrow \mathsf{KeyGen}(1^\lambda)$ is the key generation algorithm that takes as input the security parameter 1^λ, and it outputs an embedding key ek and an extraction key dk.
- $\mathsf{st} \leftarrow \mathsf{Embed}_{\mathsf{ek}}(m)$. Given an embedding key ek, and a hidden message $m \in \{0,1\}^*$, Embed generates a stegotext message $\mathsf{st} \in \{0,1\}^*$ that is *indistinguishable* from the normal channel distribution $\mathcal{D}$.

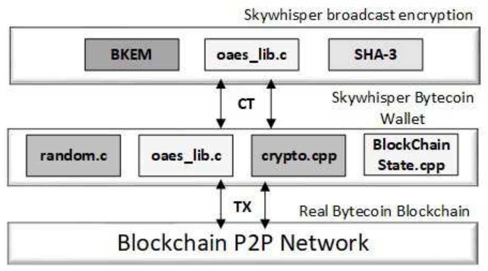


Fig. 2: Skywhisper Components.

- $m \leftarrow \mathsf{Extract}_{\mathsf{dk}}(\mathsf{st})$, Extract takes as input a extraction key dk and the stegotext $\mathsf{st} \in \{0,1\}^*$ and outputs a hidden message $m \in \{0,1\}^*$.

For definitions of stegosystems security, correctness, and reliability, please refer to [15] and [9].

Broadcast Encryption. Broadcast encryption, pioneered by Fiat and Naor [12], is a type of encryption where one ciphertext CT is transmitted in a broadcast channel to all n users in a group $\mathcal{U}$. However, only privileged users $\mathcal{S}$ can correctly decrypt it using their respective private keys. Where $\mathcal{S}$ is any subset of $\mathcal{U}$, i.e. $\mathcal{S} \subseteq \mathcal{U}$. More formally, as in [2] which we use in this work, a broadcast encryption scheme $\mathcal{BE}$ is a tuple of three algorithms $\mathcal{BE} = (\mathsf{Setup}, \mathsf{Enc}, \mathsf{Dec})$ as follows:

- $(\{d_1, .., d_n\}, \mathsf{PK}) \leftarrow \mathsf{Setup}(n)$. The setup algorithm takes the number of users n as input, and generates one public key PK and n private keys $\{d_1, .., d_n\}$.
- $(\mathsf{Hdr}, \mathsf{K}) \leftarrow \mathsf{Enc}(\mathcal{S}, \mathsf{PK})$. The encryption algorithm takes as input a subset of privileged users $\mathcal{S} \subseteq \mathcal{U}$ and a public key PK. While it outputs a header Hdr and a symmetric encryption key K. This key can be used with a secure encryption algorithm to encrypt any broadcast message.
- $\mathsf{K} \leftarrow \mathsf{Dec}(\mathsf{Hdr}, \mathcal{S}, i, d_i, \mathsf{PK})$. $\forall\ i \in \mathcal{S}$, any user with index i and private key d_i can compute the broadcast encryption key K which can be used to decrypt broadcast messages.

For definitions of correctness, collusion-resistance, and semantic security, refer to [2].

III. Our Construction

System architecture. Skywhisper is a covert broadcast communication system that can be deployed on any existing blockchain platform with randomized cryptographic primitives. The prototype is developed and tested on the real-world Bytecoin blockchain. Its general objective is to override any existing blockchain and use it for the secure and covert communication of data, and the persistent storage of arbitrary content. As illustrated in Fig. 2, Skywhisper consists of three components: **i) Skywhisper broadcast encryption**, **ii) Skywhisper Bytecoin wallet**, and **iii) Bytecoin P2P network**. The third component is responsible for providing *anonymous* connectivity among the users. Also, to run a Skywhisper node, users do not require any extra infrastructure or hardware as long as they have a Skywhisper-enhanced Bytecoin wallet/client. The following paragraphs describe the first two components of Skywhisper.

Broadcast Packet: subscription-request		
Field	**Length (Bytes)**	**Description/Value**
CMD	1	0x02
Id	1	$\mathsf{Id} \in \{1, \ldots, 63\}$
R	32	random string
Hash	2	2 LSB's of $\mathsf{Hash}(\mathsf{Hash}(d_i) \oplus \mathsf{R})$

Broadcast Packet: new-Hdr		
Field	**Length (Bytes)**	**Description/Value**
CMD	1	0x01
Auth	64	Each i^{th} byte is the LSB of the corresponding $\mathsf{Hash}(\mathsf{Hdr} \| d_i)$ where d_i is the private key of the i^{th} user.
SL	8	This is a binary encoded representation of the subscribers' list. If the i^{th} bit is set, then the corresponding i^{th} user is a subscriber
Hash	12	12 LSB's of $\mathsf{Hash}(\mathsf{K})$
Hdr	1152	length = $(A + 1) * 128$, and for 64 users, $A = 8$.

Broadcast Packet: broadcast-msg		
Field	**Length (Bytes)**	**Description/Value**
CMD	1	0x04
Encrypted msg	≤ 200	The length of the encrypted message is arbitrary in principle,but for ease of implementation it is ≤ 200.

TABLE I: Structure of Skywhisper broadcast packets.

Skywhisper broadcast encryption. This component comprises the following three algorithms: **(1)** *Broadcast Key-Encapsulation Mechanism (BKEM).* This is a modified version of the C-library [14] that implements Boneh et al.'s broadcast encryption [2]. According to this protocol, let n be the total number of users in a channel, then the size of the public key PK and Hdr is $O(\sqrt{n})$, and the size of the private keys $\mathsf{d_i}$'s is just a single element. Whenever the subscribers' list is modified, a new Hdr is generated, and a new symmetric encryption key K is derived from the updated Hdr. **(2)** *AES encryption* which is included in the Bytecoin wallet source code, and used to encrypt broadcast messages under the symmetric broadcast key K in CBC mode. **(3)** *SHA-3* is used to hash the different Skywhisper commands to ensure authenticity and integrity.

The output of the broadcast encryption component is denoted as a broadcast packet CT that is either one of three distinct types: (i) subscription-request, (ii) new-Hdr, and (iii) broadcast-msg. Detailed syntax of CT packets is shown in Table I with $n = 64$ users. After executing Skywhisper's broadcast encryption, the generated CT is taken by Skywhisper Bytecoin wallet and secretly embedded into innocuous-looking transactions.

Skywhisper wallet. This component consists of a modified version of the Bytecoin wallet (vr.3.1.1). Bytecoin is an open-source cryptocurrency project [33], and it uses the ED25519 twisted Edwards curve and CryptoNote (linkable) ring signature to sign its transactions. As shown in Sec. II, this protocol has sufficiently many uncontrolled random numbers that could be exploited to enable covert communication. The embedding process consists of the following three steps.

Step 1: embedding a broadcast packet CT *in a transaction's signature.* To covertly embed CT in a signature's random elements $\{(c_i, r_i)\}$, the modified wallet generates a synthetic $\mathsf{IV} := \mathrm{AES}_z(\mathsf{rand}\|\mathsf{CMD}\|\mathsf{Length}\|s\|0000)$ where z is a 128-bit

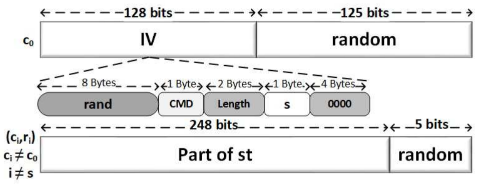


Fig. 3: Embedding a broadcast packet CT. First generate a synthetic $\mathsf{IV} := \mathrm{AES}_z(\mathsf{rand}\|\mathsf{CMD}\|\mathsf{Length}\|s\|0000)$ and embed it in c_0. Then generate a stegotext $\mathsf{st} := \mathrm{AES}_z(\mathsf{CT})$, and embed up to 31 bytes of st in each random number c_i and r_i.

channel key shared among all n users, rand is a 64-bit random string, CMD signifies one of the three packets in Table I, Length is the length of CT, s is user's secret index within the ring signature, and 0000 is a 32-bit string of 0's. Then IV is placed in the least significant 16 bytes of c_0 and sets the rest of c_0 randomly. After that, using AES-128-CBC, z and IV, the broadcaster generates a steganographic text st by encrypting CT; i.e. $\mathsf{st} := \mathrm{AES}_z(\mathsf{CT})$. Then, as illustrated in Fig. 3, the broadcaster places chunks of up to 31 bytes of st in all subsequent random numbers until the end of st. Finally, the transaction that contains st is sent as per normal over the blockchain.

Step 2: identifying transactions containing stegotext st*.* To distinguish and identify transactions containing stegotext st, receivers check every new transaction added to the ledger by decrypting the first two pairs of $(c_i,\ r_i)$ in the attached signature. In particular, a receiver uses the channel key z to decrypt the least significant 16 bytes of c_i and check if it contains 32 bits of zeros. If the receiver detects such a pattern, he identifies the existence of a stegotext and extracts IV from the least significant 16 bytes of c_i. The receiver also extracts the broadcast command CMD, the packet's length Length, and the secret index s. If, however, no such pattern is detected in any of the first two pairs of $(c_i,\ r_i)$, then the signature is ignored.

Step 3: extracting broadcast packet CT*.* After identifying the existence of st, each receiver extracts st from all random numbers. In particular, according to Length, the receiver reconstructs st by reading up to 31 bytes from all random numbers except when $i = s$. Finally, the receiver recovers the broadcast package CT by decrypting st using AEC-128-CBC, the channel key z and IV; i.e. $\mathsf{CT} := \mathrm{AES}_z^{-1}(\mathsf{st})$.

The changes introduced to the normal Bytecoin wallet affect mainly four source files: crypto.cpp, BlockchainState.cpp, oaes_lib.c, and random.c. For further clarity, the pseudo-code of Skywhisper Bytecoin wallet is shown in Fig. 4.[1]

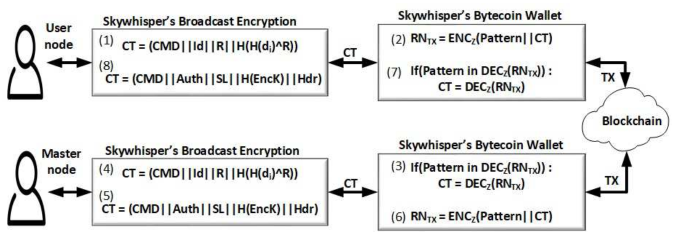
System operation. Skywhipser offers the following three main functionalities: 1) user subscription, 2) user revocation, and 3) broadcast communication.

User subscription. As illustrated in Fig. 5a, when a user wishes to subscribe to a broadcast channel the following steps occur: (1) a broadcast packet CT is created containing

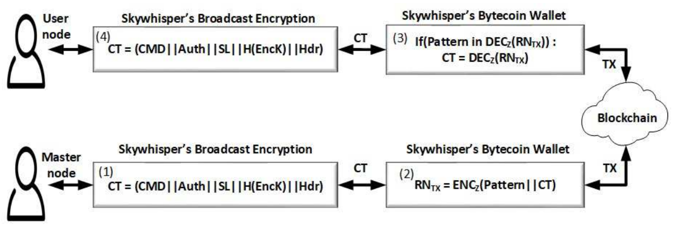
[1]For brevity, source code is not given here, but can be made available upon request.

```
function KeyGen($1^{128}$):
  • Pick random $z \leftarrow \{0,1\}^{128}$;
  • Return ek := dk := $z$;
end function
Embed$_z$(CT):
function generate_ring_signature():
  • counter = 0;
  • if(i ≠ s):
      – $c_i$ := random_scalar(s, counter + +);
      – $r_i$ := random_scalar(s, counter + +);
  • Else:
      – process as per normal;
end function
function random_scalar($s$, $counter$):
  • rand $\leftarrow \mathbb{Z}_p$;
  • if(counter == 0):
      – IV := Encrypt$_z$(rand$_{[0:63]}$||CMD||Length||s||zeros);
      – st := Encrypt$_{z,\mathsf{IV}}$(CT);
      – rand$_{[0:127]}$ := IV;
      – sent = 0;
  • if(counter > 0):
      – if(sent < Length):
          ∗ rand$_{[0:247]}$ := up to 31 bytes of st;
          ∗ sent := sent + (up to 31);
  • Return rand;
end function
Extract$_z$($c, r$):
  • pattern_found := 0;
  • for($i = 0; i < 2; i++$)
      – IV′ := Encrypt$_z^{-1}$($c_{i,[0:127]}$);
      – if(IV′$_{[96:127]}$ == zeros):
          ∗ CMD := IV′$_{[64:71]}$ Length := IV′$_{[72:87]}$ s := IV′$_{[88:95]}$ ;
          ∗ pattern := 1, index := $i$;
  • if(pattern):
      – for($i = 0; i \neq s$ & $i \neq$ index & $i < ring\ size; i++$)
      – CT := Encrypt$_{z,\mathsf{IV}}^{-1}$($c_i, r_i$);
      – Return CT;
  • Return 0;    % No broadcast message
```

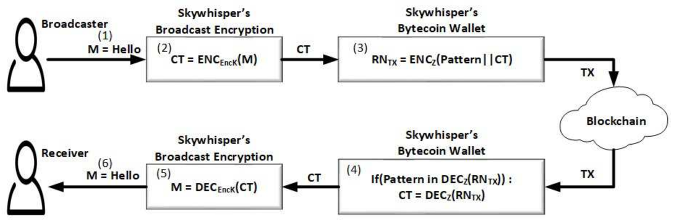
Fig. 4: Pseudo-code for the implementation of the stegosystem $\mathcal{ST}$ in the Skywhisper's modified Bytecoin wallet


User node
Skywhisper's Broadcast Encryption
(1) CT = (CMD||Id||R||H(H(di)^R))
(8) CT = (CMD||Auth||SL||H(EncK)||Hdr)
CT
Skywhisper's Bytecoin Wallet
(2) RNTX = ENCZ(Pattern||CT)
(7) If(Pattern in DECZ(RNTX)) : CT = DECZ(RNTX)
TX
Blockchain
Master node
Skywhisper's Broadcast Encryption
(4) CT = (CMD||Id||R||H(H(di)^R))
(5) CT = (CMD||Auth||SL||H(EncK)||Hdr)
CT
Skywhisper's Bytecoin Wallet
(3) If(Pattern in DECZ(RNTX)) : CT = DECZ(RNTX)
(6) RNTX = ENCZ(Pattern||CT)
TX

(a) User subscription


User node
Skywhisper's Broadcast Encryption
(4) CT = (CMD||Auth||SL||H(EncK)||Hdr)
CT
Skywhisper's Bytecoin Wallet
(3) If(Pattern in DECZ(RNTX)) : CT = DECZ(RNTX)
TX
Blockchain
Master node
Skywhisper's Broadcast Encryption
(1) CT = (CMD||Auth||SL||H(EncK)||Hdr)
CT
Skywhisper's Bytecoin Wallet
(2) RNTX = ENCZ(Pattern||CT)
TX

(b) User revocation


Broadcaster
(1) M = Hello
Skywhisper's Broadcast Encryption
(2) CT = ENCEncK(M)
CT
Skywhisper's Bytecoin Wallet
(3) RNTX = ENCZ(Pattern||CT)
TX
Blockchain
TX
Skywhisper's Bytecoin Wallet
(4) If(Pattern in DECZ(RNTX)) : CT = DECZ(RNTX)
CT
Skywhisper's Broadcast Encryption
(5) M = DECEncK(CT)
Receiver
(6) M = Hello

(c) Broadcast communication

Fig. 5: Skywhisper operations: user subscription and revocation, and broadcast communication. $\mathsf{RN}_{\mathsf{TX}}$ denotes the random elements in the ring signature.

1 byte indicating subscription-request command, the user's Id, a 32-byte random string R, and 2 LSB of Hash(Hash(d_i) ⊕ R) where d_i is the user's private key, (2) Skywhisper's Bytecoin wallet covertly embeds CT into a transaction along with a known pattern and broadcasts it through the blockchain, (3) when the master node's wallet detects the known pattern, it extracts the broadcast packet CT, (4) the master node's broadcast encryption layer authenticates the subscription request by xoring the received R with the user's secret key d_i and checking the result hash value with the received hash. If successfully authenticated, the user is added to the subscribers' list and a new Hdr is generated[2], (5) the master node generates a broadcast packet containing 1 byte indicating new-Hdr command, some authentication data Auth, the new subscribers' list SL, 12 bytes of the hash of the new encryption key K, and the new Hdr, (6) CT is secretly embedded into a transaction and transmitted through the blockchain, (7) the user node's wallet detects a transaction containing a broadcast packet and extracts CT, and (8) the user node checks the data contained in the received CT, and if it is successfully authenticated, the user node generates a new K using the received Hdr. Finally, the user node checks the correctness of K by comparing its hash to the received hash value.

User revocation. Fig. 5b shows the process of revoking the subscription of a given user. The master node initiates this process by removing the user's index from the subscribers' list and generating a new Hdr. After that, the same process is executed as steps (5)-(7) of the user subscription scenario. However, in this case, the received hash value of K is different from the computed hash value for the revoked user; hence, the user's client concludes that his subscription has been revoked.

Broadcast communication. As illustrated in Fig. 5c, any subscriber, including the master node, can send a broadcast message M to all other subscribers. The following steps are taken to broadcast M: (1) the broadcaster constructs a message M, (2) M is encrypted under the broadcast encryption key K, (3) the sender's Bytecoin wallet encrypts CT under the channel key z, embeds it into a transaction's signatures' random

[2]*Remark:* the size of Hdr is proportional to the number of users n and contains $A+1$ elements, where $n = AB$. In addition, it is stated in [2] that setting $B = \lfloor\sqrt{n}\rfloor$ gives both public key PK and Hdr of size of about $\sqrt{n}$ elements. Also, the algorithm will set up $A = \lceil \frac{n}{B} \rceil$. In our implementation of Skywhisper, we have $n = 64$, $A = 8$, and $B = 8$, which gives Hdr of 9 elements. Since each element is 128 Bytes, the size of Hdr is 1152 Bytes. After updating the subscribers' list, Skywhisper does not only broadcast Hdr to users but sends a broadcast packet CT = (CMD‖Auth‖SL‖H(K)‖Hdr) which is 1237 Bytes as further illustrated in Table I

numbers $\mathsf{RN}_{\mathsf{TX}}$, and broadcasts the constructed transaction as per normal through the blockchain P2P network, (4) each receiver's client detects a new transaction, and checks if its randomness contains the known pattern. If the pattern is found, the client extracts the received broadcast ciphertext CT and passes it to the broadcast encryption layer, (5) CT is decrypted using the broadcast encryption key K, and finally (6) the receiver correctly decrypts the broadcast message M.

IV. Evaluation of Skywhisper

Before evaluating Skywhisper, it is important to note that Skywhisper has two advantages. First, by utilizing public blockchains, *Skywhisper eliminates the need for dedicated private blockchains*. It makes use of well-established public blockchains that are backed by many nodes that participate in the consensus process incentivized by the already-available cryptocurrency. Second, *Skywhisper delegates computation outside the blockchain*. This approach removes the need for a Turing-complete programming language for the blockchain platform, uses well-tested and widely-known programming languages, and speeds up processing.

Performance. The available bandwidth in a CryptoNote transaction is given by: $\mathsf{BW} = 32(y(k-1)2)$, where y is the number of inputs in the transaction, and k is the number of public keys in each ring signature. Hence, in one transaction of 4 inputs and 10 public keys, Skywhisper can transmit more than 2KB of covert data, which, given the current price of Bytecoin is \$0.000931 [7], costs \$0.0000931. Therefore, the cost of transmitting 1MB of data via Skywhisper is ≈ \$0.05. This is much lower compared to similar techniques Tithonous [30] and R3C3 [24], where transmitting 1MB through Tithonous costs ≈ \$83, and ≈ \$6.8 in R3C3. [3]

Security and Robustness. The security of Skywhisper depends on the semantic security of the broadcast encryption scheme $\mathcal{BE}$ as defined in Sec. II, and the indistinguishability of the stegosystem $\mathcal{ST}$ implemented in the modified Bytecoin wallet to covertly embed broadcast packets CT in transactions' ring signatures. The proof of the former one can be found in [2]. Whereas, the security of the proposed stegosystem $\mathcal{ST}$ is examined for undetectability under the chosen hidden-text attacks (CHA) game/experiment. We remark that other content-insertion techniques that use non-standard Bitcoin scripts or exchange the public key with an arbitrary string with *printable* characters, as mentioned in [23][22], can be detected. However, Skywhisper's $\mathcal{ST}$ replaces random group elements in the ring signatures with pseudo-random ciphers which, by definition of semantic security, are computationally indistinguishable from truly random strings. More formally, we model the underlying encryption scheme $\mathsf{Encrypt}$ as a pseudo-random function (PRF), and we have the following theorem.

[3]Comparison is based on prices of relevant cryptocurrencies quoted on 20/05/2019 from [7].

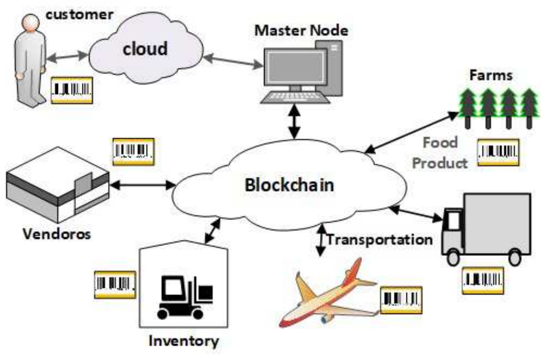


Fig. 6: Using Skywhisper for IoT communication.

Theorem 1. *If* $\mathsf{Encrypt}$ *is a secure pseudo-random function, then the stegosystem* $\mathcal{ST} := (\mathsf{KeyGen}, \mathsf{Embed}, \mathsf{Extract})$ *as shown in Fig. 4 is CHA secure.*

Proof. See App. A. □

Moreover, state-of-the-art censors can discover censorship-resistant proxies, e.g. Tor bridges, and block them. On the other hand, it is provably secure that no censors can distinguish steganographically-subverted blockchain transactions; hence, they can not launch any targeted DoS attack against Skywhisper unless they blacklist the whole blockchain network which might have other financial ramifications. Besides, other content-insertion approaches that replace segments of the transactions, as done in Tithonus [30] and Catena [34], are susceptible to policy changes where certain transactions, or scripts, become conspicuous or are no longer accepted, forcing the adoption of alternative techniques.

V. Use Cases

In this section, we explain two of the use cases of Skywhisper: IoT communication and censorship-circumvention.

Censorship Circumvention. Assuming users have access to download Skywhisper, users can use Skywhisper to evade censorship in any country where the use of the blockchain, Bytecoin in this case, is not censored.

IoT Communication. Skywhisper presents IoT communication with the following advantages:1) Skywhisper provides a ledger with immutability, non-repudiation, and persistence. 2) Skywhisper provides IoT devices with a seamless subscription and revocation process. Moreover, to illustrate the applicability of Skywhisper in storing IoT data, we apply it in one specific example; traceability of food products. According to the WHO, almost 1 in 10 people in the world fall ill after eating contaminated food [39]; therefore, food products traceability is a good case for applying Skywhisper to record IoT-generated supply-chain data in an immutable ledger [5][10][4]. In this simplified scenario, we have a *master* node, who is responsible for adding and revoking IoT devices, *farms*, *vehicles*, *stores*, and *vendors*.

In the example scenario shown in Fig. 6, the *master* node distributes offline two pairs of keys; the broadcast private key d_i and steganography channel key z, to all n potential

IoT devices. Later, when any device subscribes, as previously shown in Fig. 5a, a new broadcast encryption key K is agreed by all subscribers. In this specific example, the process begins from a *farm* which packages a given food product, assigns it a serial number SN, and broadcasts a message containing SN along with date and time dt, description text, and hash of the sender's key d_i with a random seed R, i.e. $\mathsf{M} = (\mathsf{SN}||\mathsf{dt}||\mathsf{text}||\mathsf{hash}(d_i, \mathsf{R}))$. For added security, instead of encrypting the broadcast packet CT under K, it can be encrypted under a one-time key $\mathsf{sk} = \mathsf{hash}(\mathsf{K}||\mathsf{R}||d_i)$, where R and the user's id are sent as part of the $\mathsf{Pattern}$ that is encrypted under the channel key z, i.e. $\mathsf{st} = \mathsf{ENC}_z(\mathsf{Pattern}||\mathsf{CT})$ and $\mathsf{CT} = \mathsf{ENC}_{\mathsf{sk}}(\mathsf{M})$. In this case, all subscribers can notice the existence of a broadcast message, but only the master node can decrypt the content. In a similar fashion to the producing *farm*, IoT devices along the supply chain scan the food product and generate timestamps, until the product is displayed for sale with a vendor. On the other hand, whenever the *master* node detects a broadcast message regarding a given food product, it updates that product's record in the cloud. Finally, a customer/buyer can trace the food product and check its authenticity/quality by querying the cloud using the serial number SN on the package.

VI. Related Work

This work is closely related to the following topics. **1) Censorship circumvention.** Most censorship circumvention systems fall within the following three categories. *I) End-to-end proxy-based* systems, like Tor [35], VPNs [27], and Ultrasurf [36], that establish a secure tunnel between the censored user and a proxy outside the censored area. As censors actively scan and block IP addresses, many countermeasures have been proposed among which is DEFIANCE [20] that makes it more difficult for censors to scan Tor bridges. *II) Decoy routing* in which users generate steganographically tagged traffic, so that decoy proxies can identify and redirect their traffic from its overt destinations to its actual covert destinations [16][26][40]. *III) Imitating uncensored protocols* such as StegoTorus, which disguises Tor packets in an innocuous cover protocol to evade protocol analysis of Tor packets [37], and Skypemorph [21]. **2) Blockchain-based censorship circumvention.** Tithonus [30], for example, offers a Bitcoin-based censorship-resistant system. Also, Catena [34] is an application that uses Bitcoin OP_RETURN transactions to establish consensus among users on an application-specific log. The more recent work of Minaei et al. [24] presented a Zcash-based censorship-bootstrapping tool called R3C3, and explored content insertion techniques in other cryptocurrencies. **3) Blockchain and IoT.** Among the myriads of proposals to manage IoT devices using blockchain, Novo presented a blockchain-based architecture for access management in IoT [28], similarly Pinno et al. proposed the use of blockchain for access control in IoT [29], while Huh et al. proposed the use of blockchain to configure and manage IoT devices, and built a proof-of-concept system on Ethereum [17].

VII. Conclusion and Future Work

In this work, we presented Skywhisper – the world's first covert broadcast communication system. Although we demonstrated the implementation of Skywhisper on Bytecoin, the same principles apply to any blockchain platform with randomized cryptographic primitives. Unlike dedicated and permissioned blockchain applications, Skywhisper offers the advantage of using well-established public blockchains that have many nodes participating in its consensus incentivized by the already available crypto coins. In short, Skywhisper provides a proof-of-concept solution in which a public blockchain platform can simultaneously be utilized for censorship-circumvention, IoT communication, and other applications. In general, the blockchain platform should provide enough bandwidth in its transactions to enables these various applications. In return, the blockchain platform would increase its user base and potentially increase its coin's market value.

References

[1] Kevin Ashton. That 'internet of things' thing. Available Online: https://www.rfidjournal.com/articles/view?4986 (Last accessed 02-Aug-2019).

[2] D. Boneh, C. Gentry, and B. Waters. Collusion resistant broadcast encryption with short ciphertexts and private keys. In Victor Shoup, editor, *Advances in Cryptology – CRYPTO 2005*, pages 258–275, Berlin, Heidelberg, 2005. Springer Berlin Heidelberg.

[3] Bytecoin Org. Bytecoin (BCN). Available Online: https://bytecoin.org/ (Last accessed 23-May-2019).

[4] David Cassel. Walmart's blockchain program may transform the way we use data. Available Online: https://thenewstack.io/walmarts-blockchain-program-may-transform-the-way-we-use-data/ (Last accessed 02-Aug-2019).

[5] Sylvain Charlebois. How blockchain technology could transform the food industry. Available Online: http://theconversation.com/how-blockchain-technology-could-transform-the-food-industry-89348 (Last accessed 02-Aug-2019).

[6] Civic Technologies Inc. Civic whitepaper, 2017. Available online: https://tokensale.civic.com/CivicTokenSaleWhitePaper.pdf, (Last accessed 03-Aug-2019).

[7] CoinMarketCap. Cryptocurrency market capitalizations. Available Online: https://coinmarketcap.com/ (Last accessed 22-May-2019).

[8] CryptoNote Org. CryptoNoteCoin. Available Online: http://cryptonote-coin.org/ (Last accessed 23-May-2019).

[9] Nenad Dedić, Gene Itkis, Leonid Reyzin, and Scott Russell. Upper and lower bounds on black-box steganography. *Journal of Cryptology*, 22(3):365–394, Jul 2009.

[10] Kevin Drum. Blockchain's latest triumph: Mango tracking. Available Online: https://www.motherjones.com/kevin-drum/2018/05/blockchains-latest-triumph-mango-tracking/ (Last accessed 02-Aug-2019).

[11] Dave Evans. The internet of things how the next evolutio of the internet is changing everything, April 2011. whitepaper, Available online: https://www.cisco.com/c/dam/en_us/about/ac79/docs/innov/IoT_IBSG_0411FINAL.pdf, (Last accessed 02-Aug-2019).

[12] Amos Fiat and Moni Naor. Broadcast encryption. In Douglas R. Stinson, editor, *Advances in Cryptology — CRYPTO' 93*, pages 480–491, Berlin, Heidelberg, 1994. Springer Berlin Heidelberg.

[13] V. Gatteschi, F. Lamberti, C. Demartini, C. Pranteda, and V. Santamara. Blockchain and smart contracts for insurance: Is the technology mature enough? *Future Internet*, 10(2), 2018.

[14] Oliver Gnther. Broadcast key encapsulation mechanism github repository, 2012. Available Online: https://github.com/oliverguenther/PBC_BKEM (Last accessed 08-May-2019).

[15] Nicholas J. Hopper, John Langford, and Luis von Ahn. Provably secure steganography. In *CRYPTO 2002*, 2002.

[16] A. Houmansadr, G.T.K. Nguyen, M. Caesar, and N. Borisov. Cirripede: Circumvention infrastructure using router redirection with plausible deniability. In *Computer and Communications Security*, CCS '11, pages 187–200, New York, NY, USA, 2011. ACM.

[17] S. Huh, S. Cho, and S. Kim. Managing iot devices using blockchain platform. In *2017 19th International Conference on Advanced Communication Technology (ICACT)*, pages 464–467, Feb 2017.

[18] Medicalchain Inc. Medicalchain whitepaper v2.1, 2018. Available online: https://medicalchain.com/Medicalchain-Whitepaper-EN.pdf, (Last accessed 03-Aug-2019).

[19] Y. Jie, J. Y. Pei, L. Jun, G. Yun, and X. Wei. Smart home system based on iot technologies. In *2013 International Conference on Computational and Information Sciences*, pages 1789–1791, June 2013.

[20] P. Lincoln, I. Mason, P. Porras, V. Yegneswaran, Z. Weinberg, J. Massar, W. Simpson, P. Vixie, and D. Boneh. Bootstrapping communications into an anti-censorship system. In *Presented as part of the 2nd USENIX Workshop on Free and Open Communications on the Internet*, 2012.

[21] H. M. Moghaddam, Baiyu Li, M. Derakhshani, and I. Goldberg. Skypemorph: Protocol obfuscation for tor bridges. In *Proceedings of the 2012 ACM Conference on Computer and Communications Security*, CCS '12, pages 97–108, New York, NY, USA, 2012. ACM.

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

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

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

[25] Satoshi Nakamoto. A Peer-to-Peer Electronic Cash System, 2008. Available Online: https://bitcoin.org/bitcoin.pdf (Last accessed 23-May-2019).

[26] M. Nasr, H. Zolfaghari, and A. Houmansadr. The waterfall of liberty: Decoy routing circumvention that resists routing attacks. In *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*, CCS '17, pages 2037–2052, 2017.

[27] D. Nobori and Y. Shinjo. VPN gate: A volunteer-organized public VPN relay system with blocking resistance for bypassing government censorship firewalls. In *Proceedings of the 11th USENIX Symposium on Networked Systems Design and Implementation (NSDI 14)*, pages 229–241, 2014.

[28] O. Novo. Blockchain meets iot: An architecture for scalable access management in iot. *IEEE Internet of Things Journal*, 5(2):1184–1195, April 2018.

[29] O. J. A. Pinno, A. R. A. Gregio, and L. C. E. De Bona. Controlchain: Blockchain as a central enabler for access control authorizations in the iot. In *GLOBECOM 2017 - 2017 IEEE Global Communications Conference*, pages 1–6, Dec 2017.

[30] Ruben Recabarren and Bogdan Carbunar. Tithonus: A bitcoin based censorship resilient system. *Proceedings on Privacy Enhancing Technologies*, 2019(1):68 – 86, 2019.

[31] Nicolas Van Saberhagen. Cryptonote v 2.0, 2013. whitepaper, Available online: https://cryptonote.org/whitepaper.pdf, (Last accessed 23-May-2019).

[32] L. Sanchez, L. Muñoz, J. A. Galache, P. Sotres, J. R. Santana, V. Gutierrez, R. Ramdhany, A. Gluhak, S. Krco, E. Theodoridis, and D. Pfisterer. Smartsantander: Iot experimentation over a smart city testbed. *Computer Networks*, 61:217 – 238, 2014.

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

[34] A. Tomescu and S. Devadas. Catena: Efficient non-equivocation via bitcoin. In *2017 IEEE Symposium on Security and Privacy (SP)*, volume 00, pages 393–409, May 2017.

[35] Tor Project Inc. Tor. Available Online: https://www.torproject.org/ (Last accessed 22-May-2019).

[36] Ultrareach Internet Corp. Ultrasurf. Available Online: https://ultrasurf.us/ (Last accessed 22-May-2019).

[37] Z. Weinberg, J. Wang, V. Yegneswaran, L. Briesemeister, S. Cheung, F. Wang, and D. Boneh. Stegotorus: A camouflage proxy for the tor anonymity system. In *Proceedings of the 2012 ACM Conference on Computer and Communications Security*, CCS '12, pages 109–120, 2012.

[38] Gavin Wood. Ethereum: A secure decentralised generalised transaction ledger, 2017. Available online: https://gavwood.com/paper.pdf, (Last accessed 03-Aug-2019).

[39] World Health Organization. Food safety, 2019. Available online: https://www.who.int/en/news-room/fact-sheets/detail/food-safety, (Last accessed 05-Aug-2019).

[40] E. Wustrow, C. M. Swanson, and J. A. Halderman. Tapdance: End-to-middle anticensorship without flow blocking. In *23rd USENIX Security Symposium (USENIX Security 14)*, pages 159–174, 2014.

Appendix A
Stegosystem Security Proof

Proof. We prove Theorem 1 by reduction. Assume there exists a PPT adversary $\mathcal{A}$ who can break $\mathcal{ST}$ with an non-negligible $\mathsf{Adv}^{\mathsf{CHA}}_{\mathcal{A},\mathcal{ST}}(1^\lambda)$ advantage w.r.t. the CHA game. We need to construct a PPT adversary $\mathcal{B}$ who can break the PRF game for $\mathsf{Encrypt}$. During the reduction game, $\mathcal{B}$ plays as a challenger for $\mathcal{A}$ in the CHA game. Upon receiving m from $\mathcal{A}$, $\mathcal{B}$ picks random $\mathsf{rand} \leftarrow \{0,1\}^{64}$ and sets $x := \mathsf{rand}\|00\ldots0$. $\mathcal{B}$ then queries x to the PRF game challenger and obtains IV. Subsequently, $\mathcal{B}$ queries CT to the PRF game challenger, and obtains rand. $\mathcal{B}$ then compute (c,r) according to the description shown in Fig. 4. $\mathcal{B}$ flips a coin $b \leftarrow \{0,1\}$. If $b = 0$, $\mathcal{B}$ computes a ring signature using (c,r); otherwise, $\mathcal{B}$ computes a ring signature normally. $\mathcal{B}$ then sends the resulting signature to $\mathcal{A}$. Finally, $\mathcal{A}$ outputs a guess b'. Assume the challenge bit in the PRF game is β, i.e. $\beta = 0$ is in the PRF mode; $\beta = 1$ is in the random function mode. If $b = b'$, $\mathcal{B}$ outputs $\beta^* = 0$; otherwise, $\mathcal{B}$ outputs $\beta^* = 1$.

$$\begin{aligned}\Pr[\mathcal{B}\text{ win}] &= \Pr[\beta^* = 0|\beta = 0]\cdot\Pr[\beta = 0]+\\ &+\Pr[\beta^* = 1|\beta = 1]\cdot\Pr[\beta = 1]\\ &= \Pr\left[\mathbf{Expt}^{\mathsf{CHA}}_{\mathcal{A}}(1^\lambda)\right]\cdot\frac{1}{2}+\frac{1}{2}\cdot\frac{1}{2}\\ &= (\mathsf{Adv}^{\mathsf{CHA}}_{\mathcal{A},\mathcal{ST}}(1^\lambda)+\frac{1}{2})\cdot\frac{1}{2}+\frac{1}{4}\\ &= \frac{1}{2}\cdot\mathsf{Adv}^{\mathsf{CHA}}_{\mathcal{A},\mathcal{ST}}(1^\lambda)+\frac{1}{2}\end{aligned}$$

Hence, the advtantage of $\mathcal{B}$ w.r.t to the PRF game is

$$\mathsf{Adv}^{\mathsf{PRF}}_{\mathcal{B}} = \left|\Pr[\mathcal{B}]\text{ win} - \frac{1}{2}\right| = \frac{1}{2}\cdot\mathsf{Adv}^{\mathsf{CHA}}_{\mathcal{A},\mathcal{ST}}(1^\lambda)\ \ .$$

Since $\mathsf{Adv}^{\mathsf{CHA}}_{\mathcal{A},\mathcal{ST}}(1^\lambda)$ is non-negligible, we have $\mathsf{Adv}^{\mathsf{PRF}}_{\mathcal{B}}$ is also non-negligible, which concludes the proof. ☐