Blockchain-based Authenticated Stego-Channels: A Security Framework and Construction

Vikram Kanth
Department of Electrical and Computer Engineering
Naval Postgraduate School
Monterey, USA
vkkanth@nps.edu

Britta Hale
Department of Computer Science
Naval Postgraduate School
Monterey, USA
britta.hale@nps.edu

2022 IEEE International Conference on Blockchain (Blockchain) | 978-1-6654-6104-7/22/$31.00 ©2022 IEEE | DOI: 10.1109/BLOCKCHAIN55522.2022.00036

***Abstract*—While blockchain has been an innovative and disruptive technology across many different domains, one area in which is has proven to be particularly game changing is in steganography. Steganography offers data privacy by hiding the existence of data, a property that is distinct from confidentiality (i.e. data existence is known but access is restricted) and authenticity (data existence is known but manipulation is restricted). Combinations of the latter two properties are common in analyses such as Authenticated Encryption with Associated Data (AEAD), yet there is a lack of research combining such guarantees with steganography. In this work, we introduce the security definition of Authenticated Stegotext with Associated Data (ASAD) which captures steganographic properties even when there is contextual information provided alongside the hidden data. Furthermore, we present a real-world stego-embedding scheme, Authenticated SteGotex with Associated tRansaction Data (ASGARD), that leverages a blockchain-based application as a medium for sending hidden data. We analyze ASGARD in our framework, demonstrating that it meets Level-4 ASAD security.**

***Index Terms*—Blockchain, steganography, Ethereum, covert channels, AEAD, authenticated hidden text**

I. Introduction

Information concealment in existing communications channels has recently become a topic of increased interest. There are three major ways to perform this concealment [1]. The first is hiding the data content (i.e. confidentiality). Alternatively, the identities of the communicating parties can be hidden, preventing the communication from being attributed. Finally, the existence of data can be obfuscated. Into this latter category falls steganography, the art of concealing secret information inside of other communications that are not secret.

Many of the modern use cases for steganography such as evading network censorship by nation state actors [2], require more than a guarantee that a message is hidden. We expand the typical 'stego-adversary' from simply a guessing experiment to one that also has the ability to forge, replay, reorder, or drop communications, in line with typical protocol analyses for establishment of secure channels. Investigating these extension directions, we adapt standard notions of Authenticated Encryption with Associated Data (AEAD) [3] to the steganographic space. We introduce the security definition of Authenticated Stegotext with Associated Data (ASAD), which captures steganographic properties even when there is contextual information provided alongside the hidden data, and consider authentication properties in distributed ledgers for inherent support of ASAD-secure channels.

In this work we expand the view of actionable steganography through the consideration of associated data. More specifically, we introduce the notion of Authenticated Stegotext with Associated Data (ASAD), building off of prior AEAD work and expanding steganographic analysis to include contextual data, present a blockchain-based construction for stegotext Ethereum transmissions, Authenticated SteGotext with Associated tRansaction Data (ASGARD), and analyze our ASGARD construction in the ASAD model while describing real-world considerations for deployment on the Ethereum blockchain.

The paper is organized as follows. Section II provides an overview of related work, Section III covers preliminary primitives, Section IV introduces the formal definitions for a stegotext embedding scheme with associated data and security experiments for ASAD, Section V provides a stegotext embedding construction based on the Ethereum blockchain application, and Section VI provides a security overview.

II. Related Work

A. Secure Channels

Channel security has received extensive investigation, particularly with respect to confidentiality and authentication. Furthermore, in settings where communicating parties must be concerned with both aspects, authenticated encryption is a critical primitive with an associated security model that combines both aspects (Authenticated Encryption with Associated Data, a.k.a. AEAD) [4]. Additionally, applications frequently require reliable delivery of a sequence of messages, which necessitates cryptographic modeling of an adversary whose goal it is to replay, reorder, or drop messages – an extension of the AEAD guarantees [5].

While research has been devoted to such channel security variations for confidentiality and authenticity, an analysis of similar guarantees for privacy are lacking. Basic steganographic channel models exist [6], [7], but stego-equivalent notions mirroring AEAD or the hierarchy of AEAD channel types, do not. Nonetheless, there is a real-world need as communicating parties in a stego-channel have a natural concern for both the detectability and authenticity of their hidden

978-1-6654-6104-7/22/$31.00 ©2022 IEEE
DOI 10.1109/Blockchain55522.2022.00036

communications, mirroring the confidentiality and authenticity pairing. Our work addresses this challenge.

B. Steganography

The desire to hide information in an undetectable manner is not new. As noted in Section I, steganography has been practiced for thousands of years across history and across media types. While there have been several different proposed steganographic models, Simmons' seminal model in the Prisoners' Problem [8] in which two prisoners, Alice and Bob attempt to escape a prison manned by a Warden, provides a core investigation. There have been several attempts to formally define steganographic modeling [7], [9], [10]. In [7] the authors model the steganographic channel as a distribution on bit sequences where each bit is also timestamped using increasing time values. The stego-security goal of Alice and Bob then becomes the production of stegotexts that are indistinguishable from the channel distribution. Furthermore, they introduce the concept of Chosen Hiddentext Attack (CHA) in which the Warden is allowed to provide hiddentexts to Alice. We leverage an extension of this model in Section IV. Unfortunately, the existing steganographic models to not take into account other potential avenues of attack for the Warden. There are 4 key attributes that a secure steganographic model would need to exhibit [10]: no participating party is able to forge messages that would be considered authentic, all parties are able to independently verify the authenticity of messages, no unilateral action by any party is able to impact the authenticity of the message, and no part of the message is concealed from the host (Warden). These attributes are analogous to the various AEAD authentication properties in the context of the Warden's ability to forge, replay, reorder, or drop communications.

Although this issue seems self-apparent and forgeries were described as part of the necessary stego-channel goals, to the best of our knowledge a framework for assessing the combined steganographic and authenticity security of a channel has not been investigated to date. Our work thus extends on that of [5] and [7] formalizing steganographic modeling as indistinguishability from the channel distribution and integrating not only associated data, but also a hierarchy of channel considerations.

C. Blockchain-based Covert Channels

Many of the same properties that make blockchain-based protocols attractive, specifically the existence of an immutable public ledger, also make them ideal candidates for steganography. We restrict our attention to permissionless blockchains, for the reason that they provide an immutable public ledger that can be contributed to and read by any actor. Distributed ledger consensus, as provided by Byzantine agreement, is important to this work since, if the Warden can alter blocks and transactions, they are able to disrupt the underlying steganographic channel used by the prisoners. Furthermore, we leverage the blockchain attribute that transactions cannot be altered or removed once they are submitted.

Blockchain has been looked at as an application case for covert channels. *MoneyMorph* [11], is one proposed *stego-bootstrapping* scheme for such channels. In that research, the authors present a number of provably secure embedding schemes in Bitcoin, Zcash, Monero, and Ethereum. In contrast, in this work we build a more generic framework (ASAD) that can be used both for stego-schemes that utilize blockchain as well as those that do not. We provide a proof-of-concept construction that illustrates specific characteristics, of both the schemes and blockchain usage, that enables it to reach the strictest form of ASAD. Thus, our work serves as a bridge between existing steganographic theory and the types of practical instantiations such as proposed in [11].

In [12], the author uses an address embedding scheme with least significant bit embedding and analyzes it using the CHA construct [7]. The work is notable in that it applies provable security as well as an ideal blockchain approach. Likewise, the scheme was meant as a proof-of-concept rather than a practical scheme. While the advantages of using blockchain are noted, their value in the greater steganographic context are not explored. There has also been significant work in covert embedding in the internals of digital signatures in blockchain protocols such as [13] and [14]. In contrast to the mentioned works, a major contribution of this paper is presentation of a framework by which steganographic approaches can be generically evaluated.

III. Cryptographic Primitives

Several cryptographic primitives and their security properties are relevant to the creation of our embedding scheme.

We borrow standard definitions for a symmetric encryption scheme Π = (Kgn, Enc, Dec), hash function $\mathrm{H}(x)$, and digitial signature scheme (Kgen, Sign, Vfy). Furthermore, we leverage the associated standard security notions of IND-CPA, pseudorandomness of H, and Strong Unforgeability under Adaptive Chosen-Message Attacks (SUF-CMA). Due to space restrictions, these are included in the full version.

IV. Steganography with AD

As introduced in Section II, the steganographic game is played between a sender, receiver, and a Warden (monitor) [8]. The sender's goal is to send information to the receiver via an open communications channel without the knowledge of the Warden. The security of their communication is measured by the adversary's ability to distinguish between a normal message and a stego message. We adopt the steganographic channel definition proposed by Hopper, Langford, and von Ahn [6] and its extension [7]. They define the communications channel as a history-dependent series of channel data that forms the basis for distribution of messages on the channel. The first model is the Current History Model in which the sender (and the relevant embedding function) can draw samples from the channels in only the current history. In the Look-Ahead model, the sender can sample based on a distribution from any history including that of future messages (see [15] for more details).

We extend the stego-embedding definition for consideration of associated data. Such data may be required for the communications channel and part of typical correspondence (e.g. contextual data for hiding messages) while not being specifically part of the stegotext. For example, associated data might be header information that must be sent authentically with the stegotext.

a) Look-Ahead Model [6]: A communication channel is modeled as a distribution on timestamped bit sequences. The formalization is as follows:

$$(\{0,1\},t_1),(\{0,1\},t_2),\ldots \text{ where } \forall i>0 : t_{i+1} \geq t_i \ .$$

There exists an oracle that is capable of drawing from the channel, i.e. to model an individual communicating on the channel as a drawn history of the timestamped bit sequences. Let the drawn channel history of timestamped bits be denoted as h, and let $\mathcal{C}_h$ be the channel distribution conditioned on h. In [6], the oracle could only sample from the current history, or in other words, the history of bits transmitted on the wire. As more bits are transmitted, the history h updates accordingly, with sample space distribution $\mathcal{C}_{h_1 \circ \ldots \circ h_n}$ based on $h = h_1, h_2, \ldots, h_n$ as incremented. We follow the extension of [15], where the oracle can draw not only from the existent channel history h but also from a distribution based on a selection of any future channel history $h = h_n, h_{n+1}, \ldots, h_l$.

b) Embedding Scheme: An embedding scheme consists of a tuple of algorithms defined in the following manner building off the definition in [16]. As noted in Section I, we extend this to cover the presence of associated data.

***Definition* IV.1.** *An embedding scheme defined as* $\Pi = (\text{Kgn}, \text{Embed}_{\mathcal{S}}, \text{Decode}_{\mathcal{S}})$ *consists of three algorithms where* $\mathcal{K}$ *is the stego-key space,* $\mathcal{M}$ *is the message space,* $\mathcal{AD}$ *is the associated data space,* $\mathcal{S}$ *is the stego-covertext channel distribution, and* h *is a channel history:*

- $\text{Kgn}(1^\lambda) \xrightarrow{\$} k_{emb}$*: A possibly probabilistic key generation algorithm that takes as input* 1^λ *where* λ *is the security parameter, and outputs a key,* $k_{emb} \in \mathcal{K}$*.*
- $\text{Embed}_{\mathcal{S}}(k_{emb}, \texttt{ad}, m, h) \xrightarrow{\$} s$*: A possibly probabilistic algorithm that takes as input a key* $k_{emb} \in \mathcal{K}$*, associated data* $\texttt{ad} \in \mathcal{AD}$*, a message* $m \in \mathcal{M}$*, a channel history* h *and outputs a stegotext* $s \in \mathcal{S}$*.*
- $\text{Decode}_{\mathcal{S}}(k_{emb}, \texttt{ad}, s) \rightarrow (\texttt{ad}, m, \alpha)$*: An algorithm that takes as input a key* $k_{emb} \in \mathcal{K}$*, associated data* $\texttt{ad} \in \mathcal{AD}$*, and a stegotext* $s \in \mathcal{S}$*, and outputs a tuple (*$\texttt{ad}, m, \alpha = 1$*) corresponding to successful message receipt, or a tuple (*$\perp, \perp, \alpha = 0$*) corresponding to failed message receipt .*

Remark 1. *While not featured in our definitions, there are communications channels in which the decode algorithm would require access to the same channel history as the encode algorithm. For example, in an ML-based model for embedding data, both the sender and receiver would be required to use the same channel history.*

An embedding scheme Π satisfies correctness except with negligible probability if, for all $k_{emb} \in \mathcal{K}$, for all $\texttt{ad} \in \mathcal{AD}$, for all $m \in \mathcal{M}$, and for all channel histories,

$$\Pr\Big[\text{Decode}_{\mathcal{S}}(k_{emb}, \texttt{ad}, \text{Embed}_{\mathcal{S}}(k_{emb}, \texttt{ad}, m, h)) = m\Big] = 1 - \text{negl}(\lambda) \ .$$

Unlike with encryption that requires an absolute decryption for correctness, stego channels account for channel noise and therefore correctness requires a bound of $1 - \text{negl}(\lambda)$ (see e.g., [6] [16] for more details).

A. Stego Indistinguishability Security

Given two transactions, one with embedded data and one without, as well as access to both an embedding oracle Embed and a sampling oracle S over the channel distribution, the adversary should not be able to determine which transaction contains embedded data with better than $\frac{1}{2}$ probability. We borrow a definition for this concept from [16] called security against Chosen Hiddentext Attacks (IND-CHA), which is based on symmetric encryption definitions like those from [17]. Let $\Pi = (\text{Kgn}, \text{Embed}_{\mathcal{S}}, \text{Decode}_{\mathcal{S}})$, λ be the security parameter, and $\mathcal{S}$ be the covertext channel distribution.

***Definition* IV.2.** *Let* $\mathcal{A}$ *be a PPT adversary against the indistinguishability of a steganographic scheme* $\Pi = (\text{Kgn}, \text{Embed}_{\mathcal{S}}, \text{Decode}_{\mathcal{S}})$ *with access to an embedding oracle* $\mathsf{Embed}(k_{emb}, \cdot, \cdot, \cdot)$*. We say that the steganographic scheme provides* Indistinguishability against Chosen Hiddentext Attacks *(is* IND-CHA*-secure) if for all such PPT adversaries* $\mathcal{A}$*, and keys* $k_{emb} \xleftarrow{\$} \text{Kgn}(1^\lambda)$ *we have that*

$$\Pr\Big[\mathcal{A}^{\mathsf{Embed}(k_{emb},\cdot,\cdot,\cdot)} = 1\Big] - \Pr\Big[\mathcal{A}^{\mathsf{S}(k_{emb},\cdot,\cdot)} = 1\Big] \leq \text{negl}(\lambda) \ .$$

where $\mathsf{S}(k_{emb}, \cdot, \cdot)$ *is a random sampling oracle that samples based on* $\texttt{ad}$ *as the second input and from* $\mathcal{S} = \mathcal{C}_h$*, for* h *provided as the third input.*

B. Authenticated Stegotext Associated Data (ASAD) Security

As noted in Section II, there are a number of other features that a steganographic scheme might provide. Many of those deal with the authenticity of the embedded message. In a perfect world, the steganographic scheme would detect or prevent forgeries, replays, reordering, and drops – i.e. authentication guarantees in addition to the privacy guarantees. A useful analogy to consider is the aforementioned AEAD, where in addition to preventing adversary *read* capabilities of the data, any change is also detected. For stegotext, this means that if an adversary (Warden) makes changes all transmitted messages to corrupt the reliability of a possible covert channel despite not knowing for certain if covert information is transmitted or not, the receiver can detect the modification. Furthermore, even if the covert communications are uncovered, e.g. by an insider threat, the guarantee of data authenticity holds.

Many AEAD security notions are relevant in steganography, where context data could affect the privacy of the stegotext and authenticity matters. We call this Authenticated Stegotext with

Associated Data (ASAD). We present a hierarchy of ASAD security notions for a stegotext embedding scheme, based on the AEAD hierarchy of [5]. The hierarchy is organized from lowest level of protection (privacy with protection against forgeries) to the highest level (privacy with protection against forgeries, replays, reordering, and drops).

Note that while we focus on authenticity combined with chosen hiddentext security, a similar hierarchy can be created as an extension for combining the triad of privacy, authenticity, and confidentiality. For example, the ASAD integrated with IND-CPA security could strengthen security requirements.

***Definition* IV.3.** *Let* Π *be an embedding scheme, let* $\mathcal{A}$ *be an PPT adversary algorithm, and let* $i \in \{1, \ldots, 4\}$ *and let* $b \in \{0, 1\}$. *The embedding experiment for* Π *with authentication condition* cond_i *and bit b is given by* $\mathrm{Exp}^{\mathsf{asad}_i\text{-}b}_{\Pi,\mathcal{A}}$ *in Figure 1. We define*

$$\mathbf{Adv}^{\mathsf{asad}_i}_{\Pi}(\mathcal{A}) = \left| \Pr\left[\mathrm{Exp}^{\mathsf{asad}_i\text{-}1}_{\Pi,\mathcal{A}}(\lambda) = 1\right] - \Pr\left[\mathrm{Exp}^{\mathsf{asad}_i\text{-}0}_{\Pi,\mathcal{A}}(\lambda) = 1\right] \right| .$$

We say that a stego-scheme $\Pi = (\mathrm{Kgn}, \mathrm{Embed}_{\mathcal{S}}, \mathrm{Decode}_{\mathcal{S}})$ *is* asad_i*-secure if* $\mathbf{Adv}^{\mathsf{asad}_i}_{\Pi,\mathcal{A}}(\lambda) \leq \mathrm{negl}$ *for some negligible function* negl *in* λ.

Adversary Win Conditions:
The following winning conditions for the adversary are adapted from [5]: cond_1 captures basic authentication; cond_2 captures basic authentication of stegotexts with no replays; cond_3 captures basic authentication of stegotexts with no replays and strictly increasing receipt; cond_4 captures basic authentication of stegotexts with no replays, strictly increasing receipt, and no drops. Each of the authenticated stegotext with associated data notions implies the level of security below it, for example security at Level 2 implies security at Level 1, etc. This relationship is formalized in the following theorem and is analogous to the analysis presented in [5].

Theorem 2. *Level-*$(i + 1)$ *ASAD implies Level-*i *ASAD. Let* $\Pi = (\mathrm{Kgn}, \mathrm{Embed}_{\mathcal{S}}, \mathrm{Decode}_{\mathcal{S}})$ *be an authentication scheme and let* $i \in \{1, 2, 3\}$. *For any adversary* $\mathcal{A}$,

$$\mathbf{Adv}^{\mathsf{asad}_i}_{\Pi,\mathcal{A}}(\lambda) \leq \mathbf{Adv}^{\mathsf{asad}_{i+1}}_{\Pi,\mathcal{A}}(\lambda).$$

The proof follows similarly as in [5] and due to space requirements is left to the full version.

V. Construction

In this section, we layout the necessary elements for our specific embedding scheme construction, Authenticated SteGotext with Associated tRansaction Data (ASGARD). To set context, we first present a description of the ideal blockchain model that serves as the platform for data transfer. We then describe variables and the embedding and decoding algorithms.

A. Ideal Blockchain

We build off of an ideal blockchain model (similar to [12]) that leverages the Ethereum platform and uses elements of its transaction structure as a vehicle for sending covert data. Many of the components of Ethereum are common across the blockchain/cryptocurrency landscape, opening up further possibilities for application of this construction elsewhere. The key components of the structure that are essential to our construction are as follows: Entity identities, Entity addresses (public addresses), Transactions, and Channel Ledger.

a) Entity Identities and Addresses: The embedding construction takes advantage of the hash functions that are routinely used in blockchain public address generation [18]. One abstracted procedure for address generation in permissionless blockchain systems is:

1) generate a private key,
2) calculate the corresponding public key,
3) calculate the hash of the public key, and
4) extract some number of bits from the hash as the public address of the entity.

The proposed embedding function will mirror the method applied in Ethereum, where a 160-bit truncated hash of the public key is used as the public address of the receiver [18].

b) Transactions: Abstractly, a transaction is simply data sent from one party to another. In concrete cryptocurrency applications, a transaction may include payment information.

Figure 2 illustrates the basic composition of our proposed stegotext construction using an Ethereum-based blockchain transaction. Succinctly, the transaction fields include sender and receiver addresses, context data (such as payment information), the transaction sequence number, and a signature of the sender over the transaction. We re-purpose the receiver address as a field for embedding the stegotext, employing a similar technique as has been used in *protocol dialecting* [19].

Conceptually, identification of the correct channel is based on the sender address. Thus, Bob's public key is not expected to align to the receiver address, which allows for re-purposing of that field. Rather, communicating entities Alice and Bob share a pre-approved set of sender addresses that both sides uniquely associate to covert communications.

In addition to the field used for stegotext transmission, there are notably several fields of *associated data (*$\mathtt{ad}$*)*. Associated data is any other set of values required for the transaction. For example, in the context of a cryptocurrency, the $\mathtt{ad}$ might include the amount of money being sent from the sender to the receiver. In Figure 2, it can be seen that the sender address, context data, and sequence number comprise the $\mathtt{ad}$, all of which is authenticated in addition to the stegotext field repurposed from the receiver address. If another field was used for transmitting the stegotext, then the receiver address would become associated data. Finally, we incorporate the transaction signature present in Ethereum.

Note that while the Ethereum specification refers to seq as a "nonce", the described generation and use is in fact for a sequence number [18]. In particular, seq is set to 0 for the first transaction at an identity address $addr_I$, and incremented thereafter for each subsequent transaction at that sender address. This is true in normal networks as opposed to test networks where the starting nonce may be different.

$\underline{\text{Exp}^{\mathsf{asad}_i-b}_{\Pi,\mathcal{A}}(\lambda)}$:

1: $k_{emb} \xleftarrow{\$} \text{Kgn}(1^\lambda)$
2: $u \leftarrow 0, v \leftarrow 0$
3: $\mathsf{out\text{-}of\text{-}sync} \leftarrow 0$
4: $b' \xleftarrow{\$} \mathcal{A}^{\mathsf{Embed}(\cdot,\cdot,\cdot),\mathsf{Decode}(\cdot,\cdot)}()$
5: **return** $(b = b')$

$\underline{\text{Oracle } \mathsf{Embed}(\mathtt{ad}, m, h)}$

1: $u \leftarrow u + 1$
2: $\mathcal{S} \leftarrow \mathcal{C}_h$
3: $sent.s^{(0)} \xleftarrow{\$} \mathcal{S}(k, \mathtt{ad}, h)$
4: $sent.s^{(1)} \leftarrow \text{Embed}_{\mathcal{S}}(k_{emb}, \mathtt{ad}, m, h)$
5: **if** $|sent.s^{(0)}| \neq |sent.s^{(1)}|$ **then**
6: **return** $\perp$
7: **end if**
8: **if** $sent.s^{(1)} = \perp$ **then**
9: **return** $\perp$
10: **end if**
11: $(sent.ad_u, sent.s_u) := (\mathtt{ad}, sent.s^{(b)})$
12: **return** $sent.s_u$ to $\mathcal{A}$

$\underline{\text{Oracle } \mathsf{Decode}(\mathtt{ad}, s)}$

1: **if** $b = 0$ **then**
2: **return** $\perp$
3: **end if**
4: $v \leftarrow v + 1$
5: $rcvd.s_v \leftarrow s$
6: $(\mathtt{ad}, m, \alpha) \leftarrow \text{Decode}_{\mathcal{S}}(k_{emb}, \mathtt{ad}, s)$
7: **if** $(i = 4) \wedge \mathsf{cond}_4$ **then**
8: $\mathsf{out\text{-}of\text{-}sync} \leftarrow 1$
9: **else if** $(\alpha = 1) \wedge \mathsf{cond}_i$ **then**
10: $\mathsf{out\text{-}of\text{-}sync} \leftarrow 1$
11: **end if**
12: **if** $\mathsf{out\text{-}of\text{-}sync} = 1$ **then**
13: **return** m
14: **end if**
15: **return** $\perp$ to $\mathcal{A}$

$\underline{\text{Authentication Conditions}}$

1) Basic authenticated stego:
$\mathsf{cond}_1 = (\nexists w : (s = sent.s_w) \wedge (\mathtt{ad} = sent.ad_w))$
2) Basic authenticated stego, no replays:
$\mathsf{cond}_2 = (\nexists w : (s = sent.s_w) \wedge (\mathtt{ad} = sent.ad_w)) \vee (\exists w < v : s = rcvd.s_w)$
3) Basic authenticated stego, no replays, strictly increasing:
$\mathsf{cond}_3 = (\nexists w : (s = sent.s_w) \wedge (\mathtt{ad} = sent.ad_w)) \vee (\exists w, x, y : (w < v) \wedge (sent.s_x = rcvd.s_w) \wedge (sent.s_y = rcvd.s_v) \wedge (x \geq y))$
4) Basic authenticated stego, no replays, strictly increasing, no drops:
$\mathsf{cond}_4 = (u < v) \vee (s \neq sent.s_v) \vee (\mathtt{ad} \neq sent.ad_v)$

Fig. 1. Authenticated Stegotext with Associated Data (ASAD) Experiment, asad_i, with authentication condition cond_i for embedding scheme $\Pi = (\text{Kgn}, \text{Embed}_{\mathcal{S}}, \text{Decode}_{\mathcal{S}})$ and adversary $\mathcal{A}$.

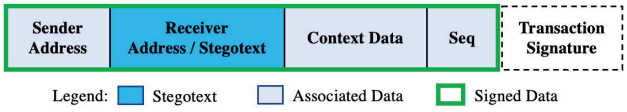

Fig. 2. Proposed Ethereum-based Transaction Structure with Stegotext

In addition, contract accounts start at one. Furthermore, the sequence number length is 64-bits [20]. The Ethereum Yellow Paper [21], as well as an Ethereum Improvement Proposal (EIP) [22] detail these distinctions in more detail.

c) Communications Ledger-Channel: Another important element of the construction that will also appear in our analysis assumptions is what we term the communications' *ledger-channel*. From a construction perspective, the ledger-channel is an immutable public ledger, similar to those realized by existing blockchain-based networks e.g. Bitcoin and Ethereum. From a security modeling perspective, this means that Alice, Bob, and Warden can all see and verify every transaction that has been sent on the channel, and such transaction cannot be later deleted. We notate this ledger-channel record as `ledger`. Note that this implies a message is *sent* after it appears on the ledger-channel, not at time of transaction generation. This modeling choice simplifies and modularizes the analysis, such that any public ledger providing the immutable property suffices, and commitment steps for uploading a transaction to a ledger are abstracted away. Note that since Alice can see the contents of `ledger` similarly to Bob and Warden, it is possible to re-send messages that are dropped during a typical ledger commitment step. As noted in Section II, the state of the network documented in `ledger` is maintained via the Byzantine agreement consensus mechanism associated with the specific blockchain protocol.

B. Definitions

Proceeding, we now present the variables associated with the stegotext embedding scheme and the construction functions.

a) Variables:

- $M \in \{0,1\}^*$: A message to be sent
- $k \in \{0,1\}^*$: A pre-shared secret key for message encryption
- seq: A sequence number initialized to 0
- $I \in ID$: A specific sender identity from a set of pre-shared sender identities
- (pk_I, sk_I): The public and private keys associated with identity I
- $addr_I := \text{H}(pk_I)$: The address of identity I, which is defined as the hash of I's public key
- ID_{pub}: The public key associated with an identity $I \in ID$, namely $I = pk_I$. We note that $ID_{pub} \subset$ `Identities` where `Identities` is any valid identity tuple in the environment
- ID_{priv}: The private tuple associated with an identity $I \in ID$, namely $I = ((pk_I, sk_I), addr_I)$
- $\mathtt{ad} := (addr_I, context, seq)$: Associated data defined as the sender address, optional transaction context data, and the sequence number seq

For the embedding scheme, Alice and Bob must share or negotiate some pieces of information prior to the setup of the

covert communication channel. These pieces of information are the secret key k and the identity set ID. The identity set ID is the set of identities that Bob knows corresponds to Alice. We emphasize that the identity set should be made sufficiently large such that each identity is only used once.[1] An identity I corresponds to the public key and address for a unique account.

Remark 3. *We note that sender addresses should only be used once as address reuse can impact the privacy of communications [23]. This is particularly important in a steganographic context since if Alice repeatedly uses the same sender address, the Warden will be able to identify all such transactions that Alice is a part of.*

Another important aspect is the role of k_{emb}, the stegotext embedding key described in Section IV. For this construction, the embedding key denotes the transaction structure and placement of embedded information. As described in Section V-A, this embedding function uses the receiver address field as for embedded information. Receiver addresses are of the form $\{0,1\}^l$, where l is the length of the address in the specific blockchain protocol. In Ethereum, the address field is 160-bits. The distribution of the 160-bit Keccak hash output used for receiver addresses forms the basis of the $\mathcal{C}_h$ distribution.

Two additional primitives are used in our construction: a symmetric encryption algorithm and a digital signature algorithm. As an intuition, we encrypt the stego-message M using Enc and embed the appropriate l-bits of stegotext in the receiver address field. Together with the associated data, the transaction is then signed using Sign (a function already associated with the Ethereum protocol). We make an important observation that, while leveraged from Ethereum, the digital signature algorithm is not native to blockchain generically for consensus. Rather it is added to various blockchain-based protocols for a variety of purposes including transaction verification and authentication. Thus, use of Sign is still within bounds of a modular composition for a blockchain public ledger utilizing Byzantine agreement.

b) Embed Function:

What follows is an informal description of our generic address embedding algorithm. Algorithm 1 provides a more formal definition of Steps 2-4. For our padding process in Step 1, we assume a standard padding algorithm.

The embed function takes as input k_{emb} (containing the private identity set ID_{priv}, the secret key k, and the channel record `ledger`), a message M, and some associated data `ad` (inclusive of $context$ data and a sequence number seq).

1) Depending on the length of M and defining l = |address field|
   a) if $|\mathrm{M}| = l$ move to Step 2.
   b) if $|\mathrm{M}| < l$, pad M until $|M| = |l|$ and move to Step 2.
   c) if $|\mathrm{M}| > l$, split M into n sub-messages $M = m_1, m_2, \ldots, m_n$ where $|m| = l$. For each m substring of M, if $|m| = l$, proceed to Step 2. Otherwise, if $|m| < l$ perform padding and move to Step 2.

   [*Algorithm 1:*]
2) Encrypt message M using the symmetric encryption algorithm Enc and pre-shared key k. This results in an encrypted message M_e of length l.
3) Embed M_e in the receiver address field.
4) Sign the transaction with the private key sk_I and initiate a `ledger` transaction using the sender address $addr_I$, where $I \in ID$, with associated data `ad` $= (addr_I, context, seq)$.

The associated data constitutes all auxiliary data.

Algorithm 1 provides a more technical description of address embedding given a message M where $|\mathrm{M}| = l$. In the case where $|\mathrm{M}| > l$, we simply run the embedding algorithm for each substring m of M, where $|m| = l$ and pad the last substring as needed. We also assume that Alice enforces sufficient time between messages sent on `ledger` so as to ensure chronological sending on the ledger-channel.

Algorithm 1: ASGARD.Embed$_{\mathcal{S}}$ Algorithm

Embed

inputs : $(ID_{priv}, k, \texttt{ledger}), context, M$
output: $(trans, \sigma)$
$M_e \leftarrow \mathrm{Enc}_k(M)$;
$((sk_I, pk_I), addr_I) \xleftarrow{\$} ID_{priv}$;
$IDpriv \leftarrow IDpriv \setminus \{I\}$;
$trans.sending\_addr \leftarrow addr_I$;
$trans.recv\_addr \leftarrow M_e$;
$trans \leftarrow$
$(trans.sending\_addr, trans.recv\_addr, context, seq)$;
$\sigma \leftarrow \mathrm{Sign}_{sk_I}(trans)$;
`ad` $\leftarrow (trans.sending\_addr, context, seq)$;
$seq++$;
`ledger` $\leftarrow$ `ledger` $\cup \{(trans, \sigma)\}$;
return `ledger`;

c) Decode Function:

Decoding is split into two parts, an extract function (Step 1) and a decode function (Steps 2-5). Our extraction and decode function algorithms are described informally as follows and formally in Algorithms 2 and 3:

[*Algorithm 2*:]
1) Monitor the channel record, `ledger` for new transactions [*Algorithm 3*:]
2) If a transaction matches a sender address in ID, move to Step 3. Otherwise return to Step 1.

[1]Note that there are various strategies for building such a set ad-hoc, which reduces the restrictions inherent in having a fully pre-fixed set of approved identities. For example, forward key derivation off of sk can be used to build out a series of public-private key pairs and consequently identities. If Alice and Bob thus treat sk as a shared secret, implying that the digital signature provides authentication albeit not non-repudiation, then ID can be extrapolated from the shared secret.

3) Verify transaction signature.
4) Extract message bits from the receiver address field of the matched transaction.
5) Decrypt the extracted text using the pre-shared key k.
6) Return to Step 1.

Algorithm 2: ASGARD.Extract Algorithm

Decode
- **inputs:** ID_{pub}, `ledger`
- **output:** $(\mathtt{ad}, s, \sigma)$
- **foreach** transaction $(trans, \sigma)$ in `ledger` **do**
  - $(sending\_addr, recv\_addr, s, context, seq) \leftarrow trans$;
  - **foreach** $pk_I \in ID_{pub}$ **do**
    - $addr_I \leftarrow \mathrm{H}(pk_I)$;
    - **if** $trans.sending\_addr = addr_I$ **then**
      - $s \leftarrow trans.recv\_addr$;
      - $\mathtt{ad} \leftarrow (addr_I, context, seq)$;
      - **return** $(\mathtt{ad}, s, \sigma)$;

Algorithm 3: ASGARD.Decode$_{\mathcal{S}}$ Algorithm

Decode
- **inputs:** $(ID_{pub}, k), \mathtt{ad}, (s, \sigma)$
- **output:** $(\mathtt{ad}, M, \alpha)$
- $(addr_I, context, seq) \leftarrow \mathtt{ad}$;
- **foreach** $pk_I \in ID_{pub}$ **do**
  - $addr_I \leftarrow \mathrm{H}(pk_I)$;
  - **if** $addr_I \notin ID_{pub}$ **then**
    - **return** $\perp$;
- **if** $seq \neq seq_I + 1$ **then**
  - **return** $\perp$;
- $seq_I + +$;
- **if** $\mathrm{Vfy}_{pk_I}((addr_I, s, context, seq_I), \sigma) = 1$ **then**
  - continue;
- $M \leftarrow \mathrm{Dec}_k(s)$;
- **if** $M \neq \perp$ **then**
  - $\alpha \leftarrow 1$;
- **return** $(\mathtt{ad}, M, \alpha)$;

Algorithm 2 describes the extraction approach and Algorithm 3 describes the decode algorithm. Both of these are for a single sub-message M, but can be easily extended to account for multiple sub-messages and consequently message sequences. Note that there is overlap between Algorithm 2 and Algorithm 3. This is intentional, since Decode$_{\mathcal{S}}$ by definition takes in the associated data in addition to the stegotext. It thus functions on a particular channel message (whether stegotext or regular covertext), and consequently individual messages must be first extracted from the ledger-channel that contains all messages in the environment. Nonetheless, for completeness we enforce Decode$_{\mathcal{S}}$ to correctly handle messages once received, including checking valid sourcing.

VI. Security Analysis

We present a security analysis of the embedding scheme under the look-ahead model presented in Section IV. Correctness follows directly from the properties of the channel record, `ledger`, and from the properties of the embedding scheme. From Theorem 2, we have that proving Level-4 ASAD implies Levels 1–3, and therefore directly target Level-4 ASAD.

Theorem 4. *Let $\mathcal{A}$ be a PPT adversary against the* asad_4 *security of* ASGARD.Embed$_{\mathcal{S}}$*, with available q_E embedding queries. Then,*

$$\mathbf{Adv}^{\mathsf{asad}_4}_{q_E,\Pi_{\text{ASGARD.Embed}_{\mathcal{S}}},\mathcal{A}}(\lambda) \leq \frac{p}{|ID|} \cdot \Big(\mathbf{Adv}^{\mathsf{SUF\text{-}CMA}}_{q_{\text{Sign}},q_{\text{Vfy}},\Pi_{\text{Sig}},\mathcal{B}_1}(\lambda) + \mathbf{Adv}^{\mathsf{IND\text{-}CPA}}_{q_{Enc},\Pi_{Enc},\mathtt{ad},\mathcal{B}_2}(\lambda) + \mathbf{Adv}^{\mathsf{prf}}_{\Pi_H,\mathcal{B}_3}(\lambda) + \mathbf{Adv}^{\mathsf{prf}}_{\Pi_{Enc},\mathcal{B}_4}(\lambda)\Big)$$

where $q_E = q_{Enc} = q_{\text{Sign}}$, p is the number of parties on the channel, ID is the set of pre-shared sender identities associated to stegotexts, $\Pi_{\text{ASGARD.Embed}_{\mathcal{S}}}$ is the stego-embedding algorithm tuple, Π_{Enc} is the encryption algorithm tuple, Π_{Sig} is the signature algorithm tuple, and $\mathtt{ad}$ is adversarially selected associated data for $\Pi_{\text{ASGARD.Embed}_{\mathcal{S}}}$.

Proof. This proof proceeds with an adversary $\mathcal{A}$ against the ASAD experiment, $\mathrm{Exp}^{\mathsf{asad}_4}_{\Pi,\mathcal{A}}$ in the following manner:

1) A challenge bit is generated: $b \xleftarrow{\$} \{0,1\}$.
2) The challenger chooses I for computing $addr_I$, and with probability $|ID| \cdot p^{-1}$ the guess correctly matches the identity chosen by $\mathcal{A}$. If $I \notin ID$, the experiment aborts.
3) The challenger generates a symmetric key k_J for each identity in the space except I and replaces H with Enc in the computation of sender identity addresses, such that $trans.recv\_addr = H(pk_J)$ is replaced by $trans.recv\_addr = \mathrm{Enc}(k_J, pk_J)$. Thus the challenger is able to correctly simulate Embed queries for every identity on the channel (i.e. randomly selected transactions based on the channel history). The ability of the adversary to distinguish between this simulation and the real distribution is bounded by the pseudorandomness of H and Enc, namely, $\mathbf{Adv}^{\mathsf{prf}}_{\Pi_H,\mathcal{B}_3}(\lambda)$ and $\mathbf{Adv}^{\mathsf{prf}}_{\Pi_{\text{Enc}},\mathcal{B}_4}(\lambda)$.
4) The challenger answers $\mathcal{A}$'s Embed queries to I using its encryption oracle Enc and signature oracle Sign. When $\mathcal{A}$ asks a query $\mathsf{Embed}(\mathtt{ad}, M, h)$, the challenger sets $M_1 \leftarrow M$ and sets $M_0 \leftarrow I$. It then calls $\mathsf{Enc}(M_0, M_1)$, followed by $\mathsf{Sign}(\mathtt{ad}, c_b)$ on the subsequent output c_b, and returns $(\mathtt{ad}, c_b, \sigma)$ to $\mathcal{A}$, where $\mathtt{ad} = (trans.sending\_addr, context, seq)$. If $b = 1$, this corresponds to a real stegomessage being sent, whereas if $b = 0$ it corresponds to the encryption of the sender's identity, in line with the simulation.
5) We next bound the probability of $\mathcal{A}$ winning according to the condition $(u < v) \vee (s \neq sent.s_v) \vee (\mathtt{ad} \neq sent.ad_v)$, i.e. if $\mathcal{A}$ is able to forge a transaction entirely, replay, change ordering, or drop a transaction. The challenger forces incremental matching of the sequence number, n, according to the ideal blockchain model (i.e.,

the security of the ideal blockchain prohibits out-of-order messages).
Since there is incremental matching of the sequence number, there is no reordering, replays, or drops, provided that no forgery has occurred. From the success of $\mathcal{A}$ we can thus build an adversary against the unforgeability of the signature scheme. Thus we bound this win condition by $\mathbf{Adv}^{\mathsf{SUF\text{-}CMA}}_{q_{\mathrm{Sign}},q_{\mathrm{Vfy}},\Pi_{Sig},\mathcal{B}_1}(\lambda)$. This bounds the ability of $\mathcal{A}$ to win according cond_4. Therefore the remaining success of $\mathcal{A}$ in the asad_4 experiment corresponds to the success of $\mathcal{A}$ without access to Decode.

6) Suppose that $\mathcal{A}$ correctly guesses the bit b. By the success of $\mathcal{A}$ in the asad_4 experiment, the adversary must correctly distinguish between a valid stegotext, generated as described above, and one randomly sampled. However, by Step (4) above, this implies correctly distinguishing between M_0 and M_1 with access to the Enc oracle, thus we bound the probability of this success by the $\mathbf{Adv}^{\mathsf{IND\text{-}CPA}}_{q_{Enc},\Pi_{\mathrm{Enc}},\mathrm{ad},\mathcal{B}_2}(\lambda)$.

□

Corollary 5. *For $i \in \{1,2,3,4\}$,* ASGARD.Embed$_{\mathcal{S}}$ *is Level-i ASAD-secure.*

The proof follows directly from Theorem 2 and Theorem 4.

Remark 6. *Note that in the ideal blockchain environment, every stegotext-embedded transaction is "sent" once it is contained in a timestamped block. Thus the model as a ledger-channel in our construction enables protection against message dropping due to transaction sequence numbers.*

Aligning the analysis and implementation choices, the challenge for the Warden would be distinguishing between an address generated via the Keccak-256 hashing algorithm and a chosen encryption algorithm for the embedded message (i.e., Step (3) of the proof). If (a) Keccak-256 output is demonstrably random according to the NIST randomness suite [24] and (b) the encryption output is indistinguishable from random, then there is a basis for this proof step assumption. One possibility for the encryption algorithm is AES-CTR mode, where the AES pseudorandom permutation over a random key provides an output for XOR with the plaintext, resulting in a ciphertext that inherits the pseudorandomness properties.

VII. Conclusion

In this work, we investigated steganographic properties in the presence of contextual information sent with the stegotext. This both expands on prior models and is more natural to the expectations of steganography, where messages are hidden in surrounding data. Finally, we note that the proliferating use of blockchain protocols offer several unique and useful features for transmission of steganographic messages. Understanding, characterizing, and leveraging those features, pushes the bounds of traditional steganography into the modern era.

References

[1] W. Mazurczyk, S. Wendzel, and S. Zander, *Information Hiding in Communication Networks: Fundamentals, Mechanisms, Applications, and Countermeasures*. John Wiley & Sons, Inc., Feb. 2016.

[2] A. Shahbaz and A. Funk, "Freedom on the net 2021: The global drive to control Big Tech," 2021. [Online]. Available: https://freedomhouse.org/report/freedom-net/2021/global-drive-control-big-tech

[3] P. Rogaway, "Authenticated-encryption with associated-data," in *Proceedings of the 9th ACM Conference on Computer and Communications Security*, ser. CCS '02. Association for Computing Machinery, 2002, p. 98–107.

[4] M. Bellare and C. Namprempre, "Authenticated encryption: Relations among notions and analysis of the generic composition paradigm," in *Advances in Cryptology — ASIACRYPT 2000*, T. Okamoto, Ed. Berlin, Heidelberg: Springer Berlin Heidelberg, 2000, pp. 531–545.

[5] C. Boyd, B. Hale, S. F. Mjølsnes, and D. Stebila, "From stateless to stateful: Generic authentication and authenticated encryption constructions with application to tls," Cryptology ePrint Archive, Report 2015/1150, 2015, https://ia.cr/2015/1150.

[6] N. J. Hopper, J. Langford, and L. von Ahn, "Provably secure steganography," in *Advances in Cryptology — CRYPTO 2002*, M. Yung, Ed. Berlin, Heidelberg: Springer Berlin Heidelberg, 2002, pp. 77–92.

[7] L. von Ahn and N. J. Hopper, "Public-key steganography," in *Advances in Cryptology - EUROCRYPT 2004*, C. Cachin and J. L. Camenisch, Eds. Springer Berlin Heidelberg, 2004, pp. 323–341.

[8] G. J. Simmons, "The prisoners' problem and the subliminal channel," in *Advances in Cryptology*. Springer, 1984, pp. 51–67.

[9] C. Cachin, "An information-theoretic model for steganography," Cryptology ePrint Archive, Report 2000/028, 2000, https://ia.cr/2000/028.

[10] G. Simmons, "Message authentication without secrecy," in *AAAS Selected Symposia Series*, vol. 69, no. 1982, 1982, pp. 105–139.

[11] M. Minaei, P. Moreno-Sanchez, and A. Kate, ": Censorship resistant rendezvous using permissionless cryptocurrencies," *Proceedings on Privacy Enhancing Technologies*, vol. 2020, no. 3, pp. 404–424, 2020. [Online]. Available: https://doi.org/10.2478/popets-2020-0058

[12] J. Partala, "Provably secure covert communication on blockchain," *Cryptography*, vol. 2, no. 3, p. 18, 2018.

[13] F. Gao, L. Zhu, K. Gai, C. Zhang, and S. Liu, "Achieving a covert channel over an open blockchain network," *IEEE Network*, vol. 34, no. 2, pp. 6–13, 2020.

[14] T. Tiemann, S. Berndt, T. Eisenbarth, and M. Liskiewicz, "Act natural!: Having a private chat on a public blockchain," Cryptology ePrint Archive, Report 2021/1073, 2021, https://ia.cr/2021/1073.

[15] A. Kiayias, Y. Raekow, A. Russell, and N. Shashidhar, "A one-time stegosystem and applications to efficient covert communication," *Journal of Cryptology*, vol. 27, no. 1, pp. 23–44, 2014.

[16] G. Kaptchuk, T. M. Jois, M. Green, and A. Rubin, "Meteor: Cryptographically secure steganography for realistic distributions," Cryptology ePrint Archive, Report 2021/686, 2021, https://ia.cr/2021/686.

[17] M. Bellare, A. Desai, E. Jokipii, and P. Rogaway, "A concrete security treatment of symmetric encryption," in *Proceedings 38th Annual Symposium on Foundations of Computer Science*, 1997, pp. 394–403.

[18] "Ethereum accounts." [Online]. Available: https://ethereum.org/en/developers/docs/accounts/

[19] M. Sjoholmsierchio, B. Hale, D. Lukaszewski, and G. Xie, "Strengthening sdn security: protocol dialecting and downgrade attacks," in *2021 IEEE 7th International Conference on Network Softwarization (NetSoft)*. IEEE, 2021, pp. 321–329.

[20] A. Beregszaszi. (2020, Apr.) EIP-2681: Limit account nonce to 2^64-1. [Online]. Available: https://eips.ethereum.org/EIPS/eip-2681

[21] G. Wood, "Ethereum: A secure decentralised generalised transaction ledger berlin version," 2021. [Online]. Available: https://github.com/ethereum/yellowpaper,[Nov.29,2021]

[22] ——. (2016, Oct.) EIP-161: State trie clearing (invariant-preserving alternative). [Online]. Available: https://eips.ethereum.org/EIPS/eip-161

[23] M. Harrigan and C. Fretter, "The unreasonable effectiveness of address clustering," in *2016 Intl IEEE Conferences UIC/ATC/ScalCom/CBDCom/IoP/SmartWorld*, 2016, pp. 368–373.

[24] L. E. Bassham III, A. L. Rukhin, J. Soto, J. R. Nechvatal, M. E. Smid, E. B. Barker, S. D. Leigh, M. Levenson, M. Vangel, D. L. Banks *et al.*, *Sp 800-22 rev. 1a. A statistical test suite for random and pseudorandom number generators for cryptographic applications*. National Institute of Standards & Technology, 2010.