NAVAL
POSTGRADUATE
SCHOOL

MONTEREY, CALIFORNIA

DISSERTATION

FRAMEWORK FOR ANONYMIZED COVERT COMMUNICATIONS: A BLOCKCHAIN-BASED PROOF-OF-CONCEPT

by

Vikram K. Kanth

September 2022

Dissertation Supervisors: John C. McEachen
Murali Tummala

Approved for public release. Distribution is unlimited.

REPORT DOCUMENTATION PAGE			*Form Approved OMB No. 0704-0188*
Public reporting burden for this collection of information is estimated to average 1 hour per response, including the time for reviewing instruction, searching existing data sources, gathering and maintaining the data needed, and completing and reviewing the collection of information. Send comments regarding this burden estimate or any other aspect of this collection of information, including suggestions for reducing this burden, to Washington headquarters Services, Directorate for Information Operations and Reports, 1215 Jefferson Davis Highway, Suite 1204, Arlington, VA 22202-4302, and to the Office of Management and Budget, Paperwork Reduction Project (0704-0188) Washington, DC, 20503.			
1. AGENCY USE ONLY ***(Leave blank)***	**2. REPORT DATE** September 2022	**3. REPORT TYPE AND DATES COVERED** Dissertation	
4. TITLE AND SUBTITLE FRAMEWORK FOR ANONYMIZED COVERT COMMUNICATIONS: A BLOCKCHAIN-BASED PROOF-OF-CONCEPT			**5. FUNDING NUMBERS**
6. AUTHOR(S) Vikram K. Kanth			
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) Naval Postgraduate School Monterey, CA 93943-5000			**8. PERFORMING ORGANIZATION REPORT NUMBER**
9. SPONSORING / MONITORING AGENCY NAME(S) AND ADDRESS(ES) N/A			**10. SPONSORING / MONITORING AGENCY REPORT NUMBER**
11. SUPPLEMENTARY NOTES The views expressed in this thesis are those of the author and do not reflect the official policy or position of the Department of Defense or the U.S. Government.			
12a. DISTRIBUTION / AVAILABILITY STATEMENT Approved for public release. Distribution is unlimited.			**12b. DISTRIBUTION CODE** A

13. ABSTRACT (maximum 200 words)

In this dissertation, we present an information hiding approach incorporating anonymity that builds on existing classical steganographic models. Current security definitions are not sufficient to analyze the proposed information hiding approach as steganography offers data privacy by hiding the existence of data, a property that is distinct from confidentiality (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 on combinations with steganography. This dissertation also introduces 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. We develop a hierarchical framework of ASAD variants, corresponding to different channel demands. We present a real-world steganographic 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 and show that it meets Level-4 ASAD security. Finally, we implement ASGARD on the Ethereum platform as a proof-of-concept and analyze some of the ways an adversary might detect our embedding activity by analyzing historical Ethereum data.

14. SUBJECT TERMS blockchain, steganography, Authenticated Encryption with Associated Data, AEAD, Authenticated Stegotext with Associated Data, ASAD, Authenticated SteGotex with Associated tRansaction Data, ASGARD, covert channels			**15. NUMBER OF PAGES** 153
			16. PRICE CODE
17. SECURITY CLASSIFICATION OF REPORT Unclassified	**18. SECURITY CLASSIFICATION OF THIS PAGE** Unclassified	**19. SECURITY CLASSIFICATION OF ABSTRACT** Unclassified	**20. LIMITATION OF ABSTRACT** UU

THIS PAGE INTENTIONALLY LEFT BLANK

FRAMEWORK FOR ANONYMIZED COVERT COMMUNICATIONS: A BLOCKCHAIN-BASED PROOF-OF-CONCEPT

Vikram K. Kanth
Lieutenant, United States Navy
BS, United States Naval Academy, 2015
MS, Electrical Engineering, Naval Postgraduate School, 2019

Submitted in partial fulfillment of the
requirements for the degree of

DOCTOR OF PHILOSOPHY IN ELECTRICAL ENGINEERING

from the

NAVAL POSTGRADUATE SCHOOL
September 2022

Approved by: John C. McEachen
Department of Electrical and
Computer Engineering
Dissertation Supervisor
Dissertation Chair

Murali Tummala
Department of Electrical and
Computer Engineering
Dissertation Supervisor

Chad A. Bollmann
Department of Electrical and
Computer Engineering

Thor Martinsen
Department of
Applied Mathematics

Preetha Thulasiraman
Department of Electrical and
Computer Engineering

Approved by: Douglas J. Fouts
Chair, Department of Electrical and Computer Engineering

Joseph P. Hooper
Vice Provost of Academic Affairs

ABSTRACT

In this dissertation, we present an information hiding approach incorporating anonymity that builds on existing classical steganographic models. Current security definitions are not sufficient to analyze the proposed information hiding approach as steganography offers data privacy by hiding the existence of data, a property that is distinct from confidentiality (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 on combinations with steganography. This dissertation also introduces 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. We develop a hierarchical framework of ASAD variants, corresponding to different channel demands. We present a real-world steganographic 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 and show that it meets Level-4 ASAD security. Finally, we implement ASGARD on the Ethereum platform as a proof-of-concept and analyze some of the ways an adversary might detect our embedding activity by analyzing historical Ethereum data.

THIS PAGE INTENTIONALLY LEFT BLANK

Table of Contents

List of Figures

THIS PAGE INTENTIONALLY LEFT BLANK

List of Tables

THIS PAGE INTENTIONALLY LEFT BLANK

List of Acronyms and Abbreviations

AD	Associated Data
ADF	Augmented Dickey-Fuller
AEAD	Authenticated Encryption Associated Data
AIC	Akaike Information Criterion
ASAD	Authenticated Stegotext Associated Data
ASGARD	Authenticated SteGotext with Associated tRansaction Data
ASIC	Application-specific Integrated Circuit
DAG	Directed Acyclic Graph
ECDSA	Elliptic Curve Digital Signature Algorithm
HMAC	Hash-based Message Authentication Code
KPSS	Kwiatkowski–Phillips–Schmidt–Shin
LL	Log-Likelihood
LSB	Least-significant Bit
MAC	Message Authentication Code
MLE	Maximum Likelihood Estimation
PBFT	Practical Byzantine Fault Tolerance
PDF	Probability Density Function
PoW	Proof-of-Work
PoS	Proof-of-Stake

PPT	Probabilistic Polynomial Time
RSA	Rivest–Shamir–Adleman
VPN	Virtual Private Network

CHAPTER 1: Introduction

The ubiquity of the Internet and networked applications has created an environment of innovation. There is hardly a field whether it be medicine, communications, finance, the military, etc., that has not been fundamentally altered by the presence of the Internet. Unfortunately, the Internet was not designed with security in mind. As one of the "Fathers of the Internet" Vint Cerf noted, the architects of the Internet were focused on creating a working product rather than the methods that actors could use to intentionally subvert it [1]. In other words, during the development of the Internet, its architects focused on the good that interconnectedness could foster rather than focusing on insiders who would attempt to destroy the network. [1] [2] [3]

The threat posed by malicious actors is increasingly poignant today, particularly for social media organizations like Facebook and Twitter, which were originally designed to be platforms that connect individuals around the world and foster positive relationships. Those same platforms have been used to spread misinformation, lies, and propaganda. There is

[1]Portions of this chapter were previously published by IEEE [2]. Reprinted, with permission, from V. Kanth, C. Bollmann, M. Tummala, and J. McEachen, "A Novel Adaptable Framework for Covert Communications in Anonymized Protocol", 2021 IEEE Military Communications Conference (MILCOM), December, 2021. This publication is a work of the U.S. government as defined in Title 17, United States Code, Section 101. Copyright protection is not available for this work in the United States. IEEE will claim and protect its copyright in international jurisdictions where permission from IEEE must be obtained for all other uses, in any current or future media, including reprinting/republishing this material for advertising or promotional purposes, creating new collective works, for resale or redistribution to servers or lists, or reuse of any copyrighted component of this work in other works.

[2]Portions of this chapter will be published by IEEE. Reprinted, with permission, from V. Kanth and B. Hale, "Blockchain-based Authenticated Stego-Channels: A Security Framework and Construction," 2022 IEEE International Conference on Blockchain (Blockchain), August, 2022. This publication is a work of the U.S. government as defined in Title 17, United States Code, Section 101. Copyright protection is not available for this work in the United States. IEEE will claim and protect its copyright in international jurisdictions where permission from IEEE must be obtained for all other uses, in any current or future media, including reprinting/republishing this material for advertising or promotional purposes, creating new collective works, for resale or redistribution to servers or lists, or reuse of any copyrighted component of this work in other works.

[3]Portions of this chapter were previously published by Springer [3]. Reproduced with permission from Springer Nature. Material from: Kanth, V., McEachen, J., Tummala, M. (2022). Parameter Identification for Malicious Transaction Detection in Blockchain Protocols. In: Prieto, J., Partida, A., Leitão, P., Pinto, A. (eds) Blockchain and Applications. BLOCKCHAIN 2021. Lecture Notes in Networks and Systems, vol 320.

broad agreement that the status quo is not tenable and that some combination of regulations and platform changes is required. Unfortunately, trust in institutions, the media, technology platforms, and expert information more generally, has cratered in the past few years [4]. This has in turn led to the development and increasing utilization of privacy-based applications like Signal, Telegram, and Tor (internet browsing). A subset of these applications is focused on providing anonymity where the entity performing an information transaction is non-identifiable, unreachable, or untrackable [5]. Examples of this type of application are cryptocurrencies (Bitcoin, Ethereum, etc.). Like the Internet and social media, these types of applications, specifically blockchain-based technologies, such as Bitcoin or Ethereum, can facilitate nefarious activity, particularly in the areas of information hiding and steganography. The goal of our research is to understand and characterize the features of these technologies that facilitate nefarious activity in order to either leverage or prevent such effects.

1.1 The Internet and Trust

It is worth examining how the Internet has been a key cog in the erosion of public trust. This problem goes far beyond social media. A 2017 Pew Research Center report [4] summarizes the various elements that have contributed to this erosion. As noted earlier, the Internet was not built with trust-building in mind. The creators of the Internet conceptualized an information sharing system that participants would use responsibly. Unfortunately, criminals have monetized cybercrime. From identity theft, to ransomware, to theft of intellectual property, there is little doubt that cybercrime is profitable [6]. Additionally, data breaches that reveal the personal information of Internet users further reduce trust in systems. Simply put, if it is dangerous to conduct any activity online while convenience and access may force one to the Internet, they may not trust that activity will be secure. Furthermore, both a growing awareness that personal data has value and the monetization of that data by large corporations have had a significant negative impact on trust. Corporations regularly use demographic data like age, race, or sex to serve targeted ads to consumers. For example, Table 1 shows the average amount that corporations paid for data from different age demographics.

Table 1.1. Cost of Data by Age. Source: [7].

Age	Cost for Data Per Person in Dollars	Percentage of Population	Total Demographic Cost in Dollars
18-24	0.36	11.92	14,253,466.27
25-34	0.11	21.80	7,791,894.90
35-44	0.12	20.30	8,368,383.54
45-55	0.27	13.76	12,346,312.34
55+	0.05	32.22	5,272,710.83

While businesses are able to profit off of consumer data, those consumers are neither compensated for their data, nor guaranteed that their data will remain secure. The argument is simple, as people grapple with the hidden costs of massive data collection and data breaches, the convenience afforded by online systems becomes much less palatable [4]. While there are proposals to potentially compensate consumers for their data, this does not fundamentally solve the issue that people do not have control over their own data. In this type of environment, it is not surprising that there is a lack of trust in the Internet.

Perhaps the most important issue regarding trust on the Internet is the fact that one can never be quite sure of who they are communicating with. In other words, the identity of an online entity is key to ensuring public trust. This has become clear in several instances, most notably Russian interference in the 2016 U.S. election. For example, the Russian Internet Research Agency (IRA) controlled a network of automated Twitter accounts whose only purpose was to amplify existing content on Twitter [8]. Americans were fooled into believing that they were communicating with other like-minded people who shared their opinions. In fact, they were not communicating with people at all. Another example of this identity hiding was the case where Russian trolls paid African-American personal trainers in parts of the United States to provide self-defense classes for African-Americans [9]. The explicit purpose of this type of effort was to sow fear and manipulate the American people using the Internet. The personal trainers had no idea that they had been contacted by agents of a foreign government instead of interested members of their community. There are many other examples in which the inability to verify the identity of an online actor has resulted in real harm to normal people. These include online scams that steal personal information or money, fake profiles on dating sites that are used for nefarious purposes, and the increasingly common ads that say a device has been infected by some virus and insist on downloading a suspicious program in order to fix the issue. Unfortunately, as the Internet its services become more and more central to peoples' lives, the types of risks associated with identity and communication will only increase.

1.2 Addressing the Trust Deficit with Privacy-based Applications

The Internet security issue and its logical extension, the trust issue, are fiendishly difficult to solve. While there have been a number of policy and regulatory proposals that have been circulated, it is clear that these interventions, while important, do not fully solve the problem and most have not been passed into law. In contrast, there is a burgeoning market of online secure products that claim to protect individual consumers from a variety of threats including hackers, identity thieves, spying governments, and companies looking to monetize personal data. These privacy-based applications are becoming increasingly common in day-to-day life. Some examples are secure messaging applications like WhatsApp, Signal, and Telegram, any number of VPN products that promise a safe browsing experience, and a rapidly expanding number of cryptocurrencies like Bitcoin and Ethereum that provide a financial system free from government oversight [10]. While these types of products can help individuals stay anonymous and secure online, they often require an understanding of the underlying service and there are so many choices that consumers are left confused. For example, a VPN that keeps logs of a user's traffic is not really private and that user likely believes that they are free from prying eyes. More importantly, this push for privacy and anonymity on the Internet can have serious consequences.

It is an unfortunate fact that security products that protect the privacy of consumers can also be used by criminals or nefarious actors to hide evidence of their crimes. Terrorist organizations routinely use privacy-based applications including Tor, Telegram, Tails, and Bitcoin among others to plan and fund attacks [11], [12]. Figure 1.1 depicts a Bitcoin address that was used in a fundraising campaign by the al-Qassam Brigades, Hamas's military wing.

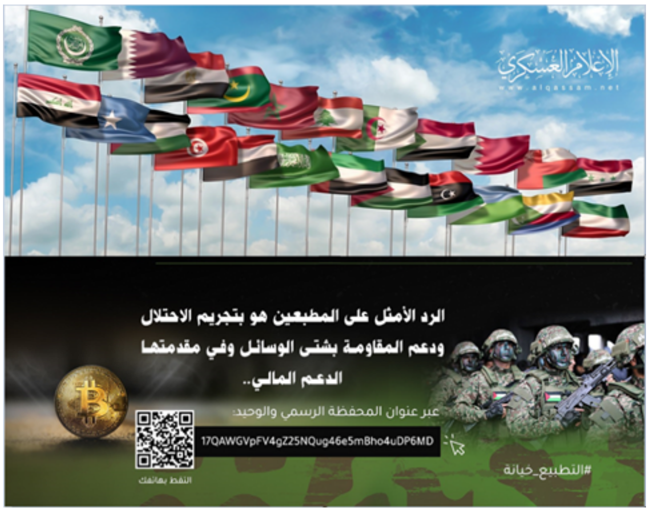
الإعلام العسكري
www.alqassam.net
الرد الأمثل على المطبعين هو بتجريم الاحتلال
ودعم المقاومة بشتى الوسائل وفي مقدمتها
الدعم المالي..
عبر عنوان المحفظة الرسمي والوحيد:
17QAWGVpFV4gZ25NQug46e5mBho4uDP6MD
التقط بهاتفك
#التطبيع_خيانة

Figure 1.1. Bitcoin Address Used by the al-Qassam Brigades. Source: [12].

It is true that law enforcement organizations have been able to successfully detect, thwart, and hold responsible malicious actors using these types of applications. For example, IRS, HSI, and FBI agents were able to track and seize 150 cryptocurrency accounts that transacted with the al-Qassam Brigades' accounts [12]. In another case, the FBI was able to arrest the leader of the Anonymous faction called "LulzSec" because he forgot to enable Tor when he visited a chat room [13]. That was enough for authorities to identify him. Many of the ways that nefarious actors use privacy-based applications are rather straightforward e.g., preventing a law enforcement agency from identifying a particular criminal actor. Unfortunately, some of these applications, particularly permissionless blockchain systems (Bitcoin and Ethereum are examples), can provide numerous unique opportunities to hide information in ways that are potentially difficult to detect. Broadly speaking, information hiding techniques utilize legitimate services to transmit hidden data. These types of techniques have been used by a variety of malicious cyber actors to support attacks and crimes [14]. It is also important to note that these types of techniques have also been used for positive efforts such as censorship avoidance [15]. Either way, it is clear that we must better understand and characterize these techniques in order to either leverage or prevent their effects.

1.3 Research Objectives

There are many research efforts that present individual approaches to hiding data in anonymized protocols. Our primary purpose is to provide a bridge between those research efforts and the underlying steganographic theory that underpins those efforts. As such, we examine how the rapid emergence of privacy-based and anonymity-based services and protocols have changed the information hiding space. This research provides a framework by which these types of techniques can be analyzed and leveraged, both from a practical perspective and in the context of provable security. We present a set of definitions and experiments that capture these security elements and are suitable for use in anonymized protocol analysis.

We then present an embedding model using blockchain as a transmission medium and analyze that model using our provable security framework, which is similar to the types of frameworks used for secure communication channels. Next, we implement our embedding model using the Ethereum blockchain and present an analysis of its security properties. Finally, we discuss methods that can be used to potentially detect that type of malicious activity by modeling Ethereum blockchain data and performing an analysis of a variety of metrics.

1.4 Related Work

Our research sought to explore the linkage between anonymity and steganographic schemes as well as the relevant security properties of those schemes. As such, we explored the fundamentals of information hiding, covert channels, and secure channels in the literature. Furthermore, the inspiration for our work was the emergence of technologies like blockchain and the unique covert communication opportunities that they presented. Thus, we also analyzed blockchain-based covert channels. The following sections present a brief overview of related work and concepts that were foundational to the development of our research.

1.4.1 Information Hiding: A Historical Overview

As the name implies, information hiding is the act of concealing a secret message within public information [16]. We remark at the beginning of this work that the terms, information hiding, information concealment, steganography, and covert channels are often used interchangeably but have subtle differences. Furthermore, the purpose of information hiding is to communicate messages between a set of entities. This message can take several different forms but can thought of as data in modern times [14]. As such, we will refer to this process of sending data between entities as a transaction.

Information hiding can be broadly split into three components. These are anonymity-based techniques, classical steganographic techniques, and cryptographic techniques [17]. Anonymity-based techniques seek to hide the identities of communicating entities. Steganographic techniques seek to obfuscate the fact that a communication is occurring typically by modifying the underlying data communication carrier such that the modification conveys the hidden message without being noticeable [17]. Cryptographic techniques seek to prevent a third party from understanding the content of a communication. These three techniques in tandem can provide for some semblance of secure communications.

The desire to communicate securely with another entity is not new at all. The first set of information hiding techniques were designed around fooling human detection using steganographic techniques. One historical example of a steganographic technique was shaving the head of a slave, tattooing the hidden message on their shaven head, waiting for their hair to regrow, and finally sending the slave to the recipient [17]. In a similar vein, corpses of animals were also used to hide messages and were carried by messengers disguised as hunters who could avoid detection [17]. These examples illustrate an important point. The selection of messenger or data carrier is a crucial element to a secure communications scheme. The messenger must not attract attention or, as Mazurczyk *et al.* concisely state, it must not be abnormal [17]. From a practical perspective, both the actual data carrying event and data carrier must be common enough to escape undue attention. For example, an exotic animal corpse would be a poor choice to hide a message in, as it would attract the attention of the local populace. Furthermore, if a nobleman was carrying an animal corpse with a message instead of a hunter, he might attract undue attention. Data carriers in steganographic applications are often called cover objects. A progression of data carriers throughout history is shown in Figure 1.2.

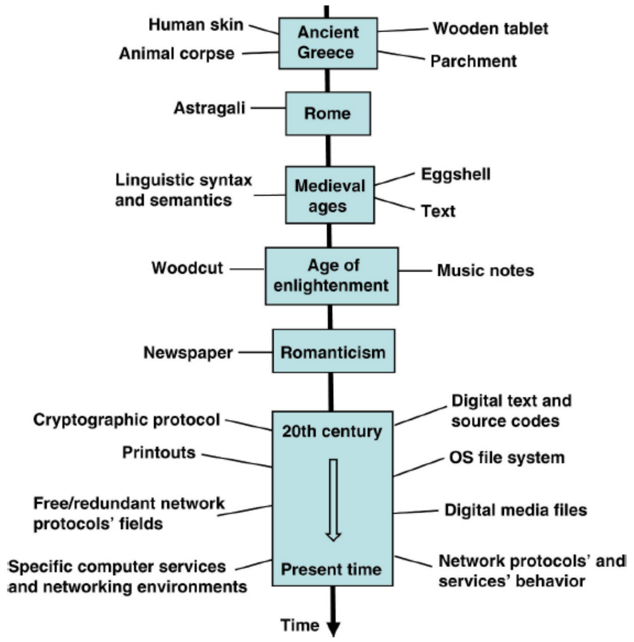

Figure 1.2. Evolution of Hidden Data Carriers Throughout History. Source: [17].

This work focuses on the interaction between anonymity-based techniques and classical steganographic techniques exploring the potential utility of new technologies, like blockchain and their unique place in information hiding efforts.

1.4.2 Covert Channels

A closely related subject to the information hiding approaches discussed in the preceding section is the covert channel. First described and defined by Lampson in his seminal 1973 work, a covert channel is a channel that is not actually intended for information transfer at all [18]. Covert channels have expanded beyond Lampson's original conception of system resources being used to illicitly communicate information to describing a whole host of approaches to hide information in a variety of communications protocols.

The most basic covert channel framework and hidden communications model is described by the well-known Prisoners' Problem depicted in Figure 1.3.

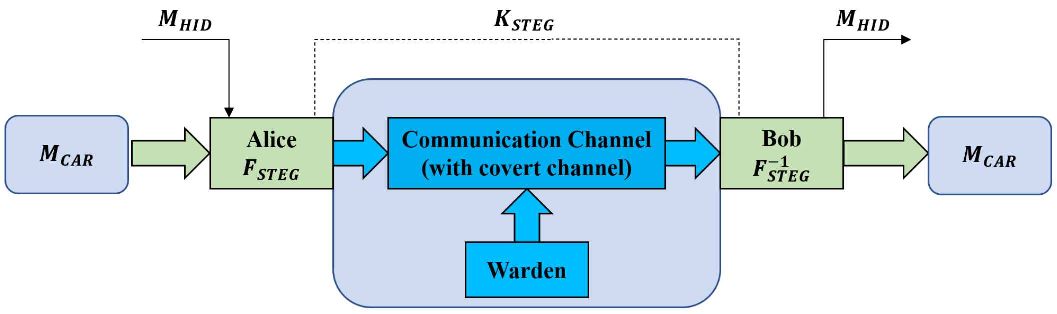

Figure 1.3. Model for Hidden Communication. Adapted from: [17].

The idea for the Prisoners' Problem was first formulated by Simmons in 1983 [19]. In this model, Alice and Bob are two prisoners that must conspire in order to hatch an escape plan. All of their methods of communication are monitored by a Warden (Note that this framework and our research assumes that an observer has access to all communications between communicating entities). If the Warden catches wind of a possible conspiracy, they punish Alice and Bob (classically by putting them in solitary confinement rendering any future escape attempt impossible). Thus, Alice must hide their hidden message (M_{HID}), in a message carrier (M_{CAR}), using a steganographic function (F_{STEG}) that can then be extracted by Bob using a steganographic function (F_{STEG}^{-1}). K_{STEG} is the shared secret between Alice and Bob that makes the hidden communication possible. We make two important remarks regarding this model. First, the message carrier is an authorized communication and not a hidden communication. The challenge of covert channels is hiding information in otherwise innocuous communications. Secondly, in order for the Warden to defeat the escape plan, all they must do is detect the presence of a hidden message, not actually be able to decipher it. The rationale for this idea is important. Imagine a scenario in which a political dissident attempts to smuggle information to a foreign reporter. If their government detects that communication effort, regardless of whether they can actually read the information, they may imprison or kill the political dissident.

It is important to note that these schemes do not take into account scenarios where the Warden is not able to positively identify Alice and Bob. Our research considers the possibility that Alice and Bob can assume multiple identities in the course of their steganographic communications. Additionally, it takes into account that certain modern protocols, such as permissionless blockchain protocols, are designed to provide that feature natively as opposed to a feature that Alice and Bob must build in. With that idea in mind, we present two of our seminal works with regards to blockchain-based covert channels.

Blockchain-based Covert Channels

At the time that this dissertation was proposed, there was only one blockchain-based covert channel that we discovered that presented an abstract embedding methodology and analysis. In [20], Partala proposes an address embedding scheme with least significant bit (LSB) embedding. He also analyzes this scheme using a rudimentary provable security approach. In contrast to his work, we present an address embedding scheme with maximal throughput (compared to one bit in Partala's approach). Though [20] notes the importance of blockchain in their steganographic scheme, its value in the greater steganographic context is not emphasized or explored further. Thus, we introduce a much more robust framework for the broad analysis of anonymity-based covert channels.

One of the most influential works that we surveyed in this area was [21] where the authors presented *MoneyMorph*, an instantiation of their proposed *stego-bootstrapping* scheme. They present a number of provably secure embedding schemes in a variety of blockchain protocols. Like our practical embedding approach, their approaches rely on blockchain protocols that use public key cryptography. In contrast, we present a steganographic framework that can be used to not only analyze their blockchain-based constructions, but also a much larger set of protocols. A key contribution of our work is that it serves as a bridge between existing steganographic theory and the types of practical instantiations such as those proposed in [20] and [21].

1.4.3 Security in a Steganographic Scheme

The fundamental question in steganography is whether a third party can detect the presence of a hidden message in an authorized communication channel i.e., can the Warden distinguish a hidden message from Alice to Bob from an authorized communication. In this context, attributes of modern media and protocols have provided unique opportunities in the steganographic space. Unfortunately, many of the modern use cases for steganography such as evading network censorship by nation state actors [15] require more than a guarantee that a message is hidden. For example, if a state-level adversary detects or even has reason to believe that a particular protocol is being used to smuggle information, it could simply block that protocol (e.g., TLS 1.3 connection blocking [22], Signal blocking [23], WhatsApp blocking [24], and DuckDuckGo blocking [25]). Thus, the conception of steganography must evolve beyond only hiding a specific message in an authorized communication channel to also considering the presence of contextual information provided alongside the hidden data. Furthermore, it is reasonable to expand the typical steganographic adversary or 'stego-adversary' from simply a guessing experiment (does an authorized message contain stego/hidden data or not) to one that also has the ability to forge, replay, reorder, or drop communications, in line with typical protocol analyses for the establishment of secure channels. Investigating these two extension directions, we adapt standard notions of Authenticated Encryption with Associated Data (AEAD) [26], [27] to the steganographic space. An important component of this dissertation introduces 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.

1.5 A Generic Information Hiding Framework

The ideas in the preceding sections provide a brief overview of the terminology and issues regarding information hiding. One of the goals of our research is to provide additional framing concepts around information hiding and concealment that extend existing ideas in the area of networked communications. Figure 1.4 depicts a basic framework capturing the broadest aperture regarding information hiding.

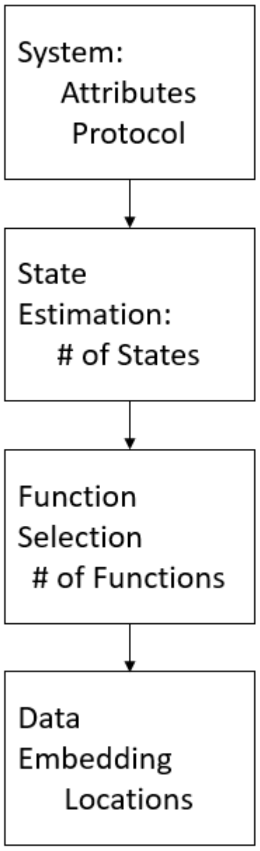

Figure 1.4. Generic Information Hiding System Framework

In the broadest sense, we start with a system with a set of attributes. These attributes could include number of participants, anonymity, etc. As will be discussed later, these attributes may provide particular avenues for information hiding. Given today's networked environment, we assert that the system is dynamic in nature. That is to say, at any given time, the state of the system can change. Every system implements some set of n protocols P that relates to the function of the system. Every protocol P_i has a set of steganographic functions F that can be utilized for communications.

Every system can be in one of m states based on defined system parameters. An example of this would be a system that had a high traffic, medium traffic, and low traffic states. An estimation of the system state would then occur. Based on that estimation and the set of functions F, an element of F, a steganographic function f would be chosen for use. Finally, specific embedding locations would be determined. This framework provides a general overview of the process required for information concealment and will be further refined throughout this dissertation.

1.6 Outline

This dissertation is organized as follows. Chapter 2 covers background information regarding steganography, provable security notions that are integral to analysis of our information hiding model, and an overview of blockchain, a technology that will be featured heavily in practical implementations of our model. Chapter 3 presents our framework for covert communications in anonymized protocols and introduces the notion of ASAD, building off of prior AEAD work and expanding steganographic analysis to include contextual data. Chapter 4 demonstrates an embedding scheme based upon the covert communications framework and ASAD model presented in Chapter 3. Chapter 5 presents a practical implementation of the embedding scheme from Chapter 4 using the Ethereum blockchain platform. An analysis of potential methods of detecting our Ethereum-based embedding model is presented in Chapter 6. In Chapter 7, we cover significant contributions from this research as well as potential avenues for future work. Finally, we provide appendices that cover miscellaneous elements of our work. Specifically, Appendix A includes the definitions and experiments for Stateful AEAD, Appendix B contains the results of the NIST randomness tests applied to our empirical data, and Appendix C is a description of code used in this work.

THIS PAGE INTENTIONALLY LEFT BLANK

CHAPTER 2: Background

In this chapter, we present an overview of a number of topics that provide context on the remainder of this dissertation. Specifically, we present an overview of steganography and some of its information theoretic models. We then discuss the topic of secure channels and the evolution of security goals from simple encryption to stateful AEAD. Following our secure channel discussion, we present an overview of blockchain and a more specific look at Ethereum internals. Finally, we talk about previous blockchain-based covert channels in the literature. [4] [5] [6]

2.1 Steganography

The desire to hide information in an undetectable manner is not new. As noted in Chapter 1, 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 [19] provides a core investigation (see Figure 1.3). To briefly summarize, Simmons poses a conundrum in which two criminal accomplices are arrested and have been thrown in jail. The Warden allows the prisoners (Alice and Bob) to exchange authorized messages. In order to escape, Alice and Bob must send messages that contain their hidden getaway plan. If they should fail, or the Warden catches wind of their efforts, they are separated indefinitely with no further escape opportunities.

The Alice/Bob/Warden problem framing allows generality in modeling the steganographic threat environment. There have been several attempts to formally define steganographic

[4]Portions of this chapter were previously published by IEEE [2]. Reprinted, with permission, from V. Kanth, C. Bollmann, M. Tummala, and J. McEachen, "A Novel Adaptable Framework for Covert Communications in Anonymized Protocols", 2021 IEEE Military Communications Conference (MILCOM), December, 2021.

[5]Portions of this chapter will be published by IEEE. Reprinted, with permission, from V. Kanth and B. Hale, "Blockchain-based Authenticated Stego-Channels: A Security Framework and Construction", 2022 IEEE International Conference on Blockchain (Blockchain), August, 2022.

[6]Portions of this chapter were previously published by Springer [3]. Reproduced with permission from Springer Nature. Material from: Kanth, V., McEachen, J., Tummala, M. (2022). Parameter Identification for Malicious Transaction Detection in Blockchain Protocols. In: Prieto, J., Partida, A., Leitão, P., Pinto, A. (eds) Blockchain and Applications. BLOCKCHAIN 2021. Lecture Notes in Networks and Systems, vol 320.

modeling. In [28], Cachin proposes an information theoretic model for steganography. Specifically, using information theoretic concepts, the author defines a perfectly secure stegosystem as one in which the Warden cannot distinguish between the covertext distribution and the stegotext distribution. Specifically, a stegosystem that is ϵ-secure against passive adversaries is defined as:

$$D(P_C \| P_S) \leq \epsilon \tag{2.1.1}$$

where P_C and P_S are the probability distributions of the covertext and stegotext respectively. The function $D(P_C \| P_S)$ is the Kullback-Liebler divergence or relative entropy defined as:

$$D(P_C \| P_S) = \sum_{x \in X} P_C(x) \log \frac{P_C(x)}{P_S(x)} \tag{2.1.2}$$

where X is the space of possible measurements. Importantly, the relative entropy is zero if and only if the distributions P_C and P_S are equal. This gives rise to Cachin's definition of a perfectly secure stegosystem, which describes an ϵ-secure stegosystem where $\epsilon = 0$. This idea will be very significant for our embedding algorithm in Chapter 4, and for our practical implementation in Chapter 5.

In [29] Hopper, Langford, and von Ahn present several formalizations for steganographic functions and crucially, model the steganographic channel as a distribution on bit sequences where each bit is also timestamped using increasing time values. This definition is covered in Chapter 3. The steganographic security goal of Alice and Bob then becomes the production of stegotexts that are indistinguishable from the underlying 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 Chapter 3.

In [19], Simmons broaches the topic of authentication in steganographic channels. Specifically, the author argues that a steganographic channel is actually a subversion of an authentication channel. The prisoners (transmitter/receiver or Alice/Bob) want to ensure that their messages came from each other. This problem is exacerbated by the fact that the Warden insists on inspecting every message's content and has the ability to manipulate messages. Simmons leverages the following four key attributes of an authentication channel [30] as the basis for building a steganographic one:

1. No participating party is able to forge messages that would be considered authentic.
2. All parties are able to independently verify the authenticity of messages.
3. No unilateral action by any party is able to impact the authenticity of the message.
4. No part of the message is concealed from the host (Warden).

While the above attributes point to a cognizance of the Warden's ability to forge or substitute communications, our discussion of regime censorship highlights the prisoners need to consider the Warden's ability to block their communications as well. In fact, the various AEAD authentication properties likewise all apply to steganographic channels – especially in the context of the Warden's ability to forge, replay, reorder, or drop communications. Thus, in order to be a useful steganographic channel, a communications channel not only must support steganographic embedding protocols, it must also provide authentication against various adversarial actions.

Although this issue seems self-apparent and forgeries were described early on as part of the necessary steganographic 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. This dissertation thus extends on that of [27] and [29] formalizing steganographic modeling as indistinguishability from the channel distribution and integrating not only associated data, but also a hierarchy of channel considerations.

2.2 Anonymity

Anonymity has been an overloaded term with several different definitions in the literature and in technology circles. We present our definition of anonymity and some of the foundational concepts regarding anonymity that are crucial in understanding our anonymized steganographic model. To explain these concepts, we borrow a problem setting from [31] where a set of senders sends messages to a set of recipients over a communications network or channel. Figure 2.1 shows a simple figure illustrating this model.

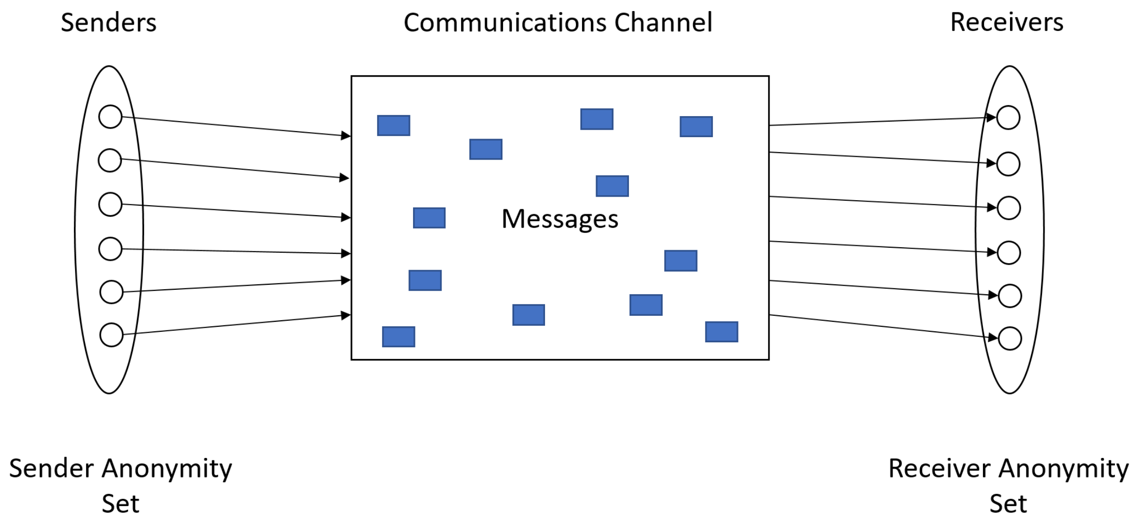

Figure 2.1. Anonymity Model. Adapted From: [31].

In this conception, the sets of possible identities that the sender and receiver can take on are referred to as their respective anonymity sets. These sets can be the same size or different sizes depending on the application. Now, we define anonymity as the property that a sender or receiver is not identifiable within their anonymity set. This follows from standard anonymity definitions that a user may perform an action without disclosing their identity (see [32]). We emphasize that the authors in [31] also provide an information theoretic definition of anonymity, which is that for a fixed anonymity set, anonymity is maximal if each identity in the anonymity set is equally likely to be used (this mirrors the notion of perfect secrecy from Shannon's seminal work [33]). This fact will be critical in our model in Chapter 3.

In this environment, the adversary represents an entity that has access to the communications channel and can participate in the communications process. Their goal is to identify senders and receivers involved in specific communications using information gleaned from observed communications. We note that the adversarial construction in this setting is similar to that of the Warden in steganographic settings. Specifically, one can model this problem in a manner similar to identifying stegotext from covertext. In this case, the adversary looks at a profile of sender/receiver activity and determines whether it matches an expected distribution of sender/receiver activity [31]. Thus, it is a logical extension to combine these concepts.

We also introduce the concepts of unlinkability and pseudonymity to round out our anonymity discussion. Unlinkability refers to the adversaries ability or lack thereof to link two or more items of interest (senders, receivers, messages, other actions) together [31]. In other words, unlinkability describes a scenario where a user can perform multiple actions without other observers being able to link those actions together [32]. Unlinkability is important in the steganographic context because a Warden linking several prisoner actions together can potentially detect an escape attempt. This idea informs some of our design decisions in Chapter 4, specifically with regards to the size of our identity set.

Finally, as we deal with blockchain and blockchain-based protocols, we define pseudonymity. Simply put, pseudonymity describes the use of a different identifier (a pseudonym) than one's own identity [31]. Common examples of this are stage names for entertainers or pen names for authors. In the cyber setting, this is an area of intense and long-term research [34] and is significant in the context of blockchain protocols like Bitcoin. In the context of Bitcoin, the use of an address is a pseudonym but does not always provide anonymity (see the example from Chapter 1 with the al-Qassam Brigades use of Bitcoin). The distinction between anonymity and pseudonymity is subtle but significant. In pseudonymity, a false name is used to disguise the identity of a party, but it might still be possible to track down the real identity of the party. Again referring to Chapter 1, government agencies were able to track people who made donations to terrorist organizations despite their use of pseudonymous cryptocurrencies. In anonymity, observers are unable to distinguish the real identity of a party conducting activities. This is why many cryptocurrencies are referred to as pseudonymous vice anonymous though there are some cryptocurrencies such as Monero [35] that were designed with anonymity in mind. In this work, we assume that our prisoners have taken appropriate steps to prevent the Warden from associating their pseudonyms with their identities. For example, if the prisoners use Bitcoin transactions to communicate, they do not use their personal information to buy the Bitcoin required for their communications.

2.3 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 [36], [37]. The goal of an authenticated encryption scheme is to provide confidentiality (an adversary cannot read the message), integrity (the message has not been changed), and authentication (the message came from the stated sender). Note that many of these works use the word authenticity, which describes both data integrity and authentication. In practice, this is commonly accomplished via message authentication codes (MACs). In [36], Bellare and Namprempre provide a comparison between three authenticated encryption composition methods. These methods are Encrypt-and-MAC, MAC-then-Encrypt, and Encrypt-then-MAC. A complete treatment of cryptographic primitives (e.g., symmetric encryption, digital signatures, etc.) and some of their relevant properties (e.g., $\mathsf{IND\text{-}CPA}, \mathsf{IND\text{-}CCA2}$ for symmetric encryption, $\mathsf{EUF\text{-}CMA}, \mathsf{SUF\text{-}CMA}$ for digital signatures) is outside the scope of this dissertation but are defined as necessary for our model in Chapter 3. We also present a brief overview of provable security and an example in Section 2.4.

With modern protocols, in addition to the message, there is other associated data (such as packet headers), that needs to be sent alongside ciphertexts authenticated, but in the clear. In other words, a protocol might have a mix of secret and non-secret data where only parts of the data need to be encrypted, but all the data must be authenticated. In [26], Rogaway defines the notion of Authenticated Encryption with Associated Data (AEAD), which captures the associated data use-case. They present nonce stealing and ciphertext translation as methods to modify an authenticated encryption scheme that does not support associated data into one that does. There are several other works that present other AEAD constructions, such as EAX from [38].

Unfortunately, basic confidentiality and authenticity properties alone are not sufficient to realize many channel security properties expected in practice. Specifically, applications frequently require reliable delivery of a sequence of messages, which necessitates cryptographic modeling of an adversary whose goal might be to replay, reorder, or drop messages. The variety of secure channel goals leads to the notions of stateful authenticated encryption and stateful AEAD [39], [40]. Further research extends classic AEAD with a hierarchy of experiments, formalizing the connections between different types of AEAD security notions [27] (See Appendix A).

While research has been devoted to channel security variations for confidentiality and authenticity, an analysis of similar privacy guarantees is lacking. Basic steganography channel models exist [29], [41], but stego-equivalent notions mirroring AEAD, or the hierarchy of AEAD channel types, do not. Nonetheless, AEAD security notions have an almost one-to-one correspondence with requirements for a secure steganographic channel. For example, communicating parties in a stego-channel have a natural concern for both the detectability and authenticity of their hidden communications, mirroring the confidentiality and authenticity pairing in a normal secure channel. For example, a specific protocol may require associated data (see for [42] examples of HTTP-based channels exhibiting this need) but also serve as an excellent vehicle for steganographic communications. Our work addresses this lack of exploration and analysis in steganographic channels.

2.4 Provable Security

One of our goals in this dissertation is to provide a framework by which the security properties of an anonymized steganographic scheme can be assessed. In order to present this framework, we present an overview of provable security and some of the related concepts required to fully understand the later chapters of our work.

The goal of the provably security paradigm is to provide some mathematical guarantee about the security of cryptosystems. To provide some historical context, the first work that delved into this subject was Shannon's seminal work [33] in which he presented a set of requirements that defined a "perfect" encryption scheme. As noted in [43], Shannon's work was difficult to extend and it was not until the introduction of asymmetric cryptography by Diffie and Hellman [44] that this area gained new life. In [45], Rabin presented the first modern security proof of a cryptosystem. Specifically, he related the difficulty of breaking a cryptosystem (in this case factoring-based public key systems like RSA) to the difficulty of solving a difficult math problem, which in this case was the factoring of a product of two large primes. Modern provable security often uses this idea of reduction in its proofs.

In this dissertation, we focus our attention on these types of reductionist proofs for game-based security definitions. We borrow a definition of a security property of a cryptographic scheme from [46] as a game between a challenger and an adversary. Security is modeled via showing that breaking the security property, i.e., winning the game, is reducible to breaking an underlying hard problem, like factoring the product of two large prime numbers.

There is a common structure in provable security models for a specific scheme like asymmetric encryption. Each model contains:

1. A scheme definition
2. A security definition, execution environment, and winning conditions
3. A class of attackers (i.e., what abilities does the attacker have in the execution environment)
4. A proof that the winning condition is unachievable given an attacker from the class of attackers previously defined [46].

Though we work with game-based security models, for completeness we present Figure 2.2 from [46], which outlines several different approaches to provable security.

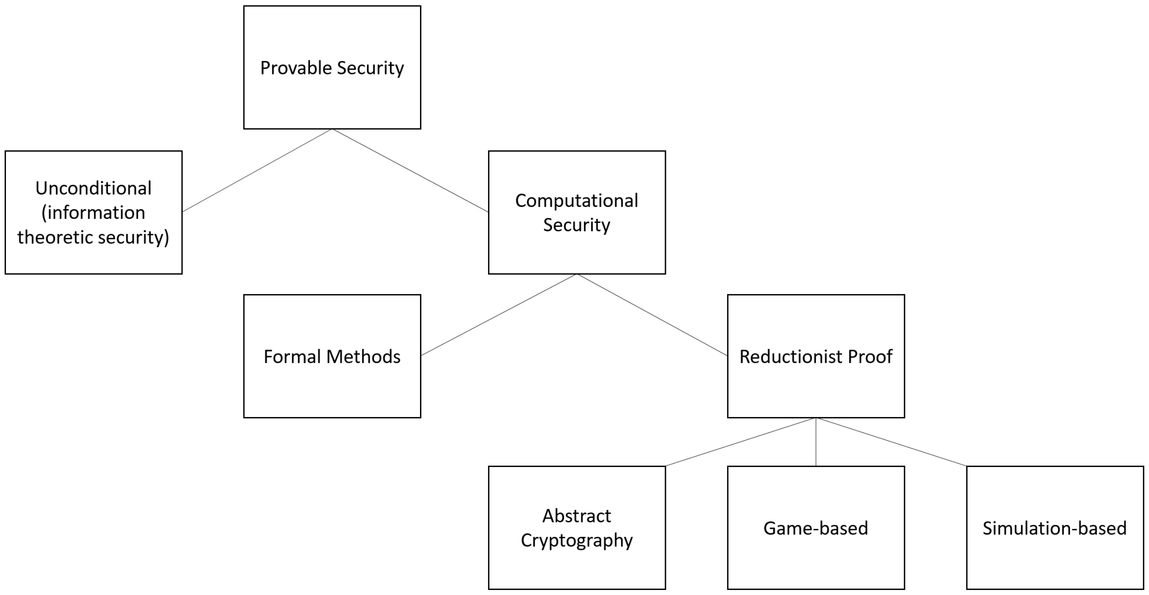

Figure 2.2. Provable Security Overview. Adapted From: [46].

We pay special attention to reductionist proofs as our game-based proofs are a subclass of them. Reductionist proofs can be either asymptotic where the adversary is probabilistic polynomial time (PPT) and their success probability should be negligible (defined here as smaller than the inverse of any polynomial, see [46]) or concrete where the adversary is probabilistic time-t and their success probability is $\leq \epsilon$. Many reductionist proofs rely on simplifying assumptions based on specific cryptographic primitives. Some examples include the random oracle model (ideal hash functions) [47] and the generic group model (ideal groups) [48].

Borrowing from [43], [46], and [49] we now present a provable security model for public key encryption as an example of the provable security paradigm.

***Definition* 2.4.1.** *A public key encryption scheme defined as* $\Pi = (\text{Kgn}, \text{Enc}, \text{Dec})$ *consists of three algorithms where* $\mathcal{M}$ *is the message space.*

- $\text{Kgn}() \overset{\$}{\rightarrow} (sk, pk)$*: A probabilistic key generation algorithm that takes no input and outputs a key pair,* sk, pk *corresponding to the secret key and public key respectively.*
- $\text{Enc}(pk, m) \overset{\$}{\rightarrow} c$*: A probabilistic encryption algorithm that takes as input a public key* pk *and a plaintext* $m \in \mathcal{M}$ *and outputs a ciphertext* c*.*
- $\text{Dec}(k, c) \overset{\$}{\rightarrow} m$ *or* $\perp$*: A deterministic decryption algorithm that takes as input a secret key* sk *and a ciphertext* c *and outputs a plaintext* m *if* c *is successfully decrypted or* $\perp$ *if there is an error in decryption.*

As part of the scheme definition for public key encryption, we must also define a notion of correctness. A public encryption scheme Π satisfies correctness if for all plaintext messages $m \in \mathcal{M}$, for all secret key/public key pairs $(sk, pk) \overset{\$}{\leftarrow} \text{Kgn}()$, and for all ciphertexts $c \overset{\$}{\leftarrow} \text{Enc}(pk, m)$,

$$\Pr\Big[\text{Dec}(sk, \text{Enc}(pk, m)) = m\Big] = 1.$$

As an important note, in some constructions, particularly steganographic constructions, the probability of correctness may not be one, but rather under some threshold value [41].

Now, we define the set of adversary goals and capabilities in the execution environment. As noted in [46], there are varied adversary goals. The adversary might be interested in recovering the secret key sk, or specifically decrypting a ciphertext, or given a ciphertext c encrypting one of two messages $m0, m1$, guessing which message resulted in c. This latter notion is known as indistinguishability or semantic security where the adversary learns nothing about the message from the ciphertext beyond the length of the text [50].

Considering the goal of indistinguishability, the adversary has a range of potential abilities each tied to the execution environment that the scheme is built in. These are well-defined in [46] as:

1. Key-only attack: The adversary has access to the public key pk.
2. Chosen-plaintext attack (CPA): The adversary also has access to an encryption black box that allows them to encrypt as many messages as they like.
3. Chosen-ciphertext attack (CCA1): The adversary also has access to a decryption black box that allows them to decrypt as many ciphertext as they like until they receive the challenge ciphertext.
4. Chosen-ciphertext attack (CCA2): The adversary now has access to a decryption black box that allows them to decrypt as many ciphertext as they like even after receipt of the challenge ciphertext (they cannot send the challenge ciphertext to the decryption black box)

A security experiment for indistinguishability under CPA (IND-CPA) proceeds as follows and is formalized in Figure 2.3.

1. The challenger computes a key pair $(sk, pk) \overset{\$}{\leftarrow} \text{Kgn}()$, and sends the public key pk to the adversary.
2. The adversary selects two messages $m_0, m_1 \in \mathcal{M}$ of the same length and sends them to the challenger.
3. The challenger picks a value for $b \in \{0, 1\}$, computes $c \overset{\$}{\leftarrow} \text{Enc}(pk, m_b)$, and sends c to the adversary.
4. The adversary picks a bit $b' \in \{0, 1\}$ and wins if $b = b'$.

```
$\underline{\text{Exp}^{\textsf{IND-CPA}}_{\Pi,\mathcal{A}}():}$

1: $(sk, pk) \overset{\$}{\leftarrow} \text{Kgn}()$
2: $(m_0, m_1, st) \overset{\$}{\leftarrow} \mathcal{A}(pk)$

3: $b \overset{\$}{\leftarrow} \{0, 1\}$
4: $c \overset{\$}{\leftarrow} \text{Enc}(pk, m_b)$
5: $b' \overset{\$}{\leftarrow} \mathcal{A}(st, c)$
6: **return** $(b = b')$
```

Figure 2.3. IND-CPA Experiment. Adapted From: [46]

The variable *st* refers to state information the adversary might use in the later guessing state (for more information see the section on find-then-guess security [51]). We note that this experiment is not the only way to show IND-CPA but provides a base primer to understand our notation later in this dissertation. Of course, in this experiment, the adversary has a $\frac{1}{2}$ probability of guessing b correctly. Thus, our concern is how much better than a random guess that the adversary can do. This quantity is known as the adversary's advantage and is defined as follows for our IND-CPA experiment:

***Definition* 2.4.2.**

$$\mathbf{Adv}^{\mathsf{IND\text{-}CPA}}_{\Pi,\mathcal{A}}() = \left|\Pr\left[\mathrm{Exp}^{\mathsf{IND\text{-}CPA}}_{\Pi,\mathcal{A}}() = 1\right] - \frac{1}{2}\right|$$

Finally, we provide a definition for a public-key encryption scheme that is secure against CPA.

***Definition* 2.4.3.** *A public key encryption scheme* $\Pi = (\mathrm{Kgn}, \mathrm{Enc}, \mathrm{Dec})$ *is secure against CPA if, for all PPT adversaries* $\mathcal{A}$*, there is a negligible function* negl *such that*

$$\mathbf{Adv}^{\mathsf{IND\text{-}CPA}}_{\Pi,\mathcal{A}}() \leq \mathrm{negl}()\ .$$

There are many other types of algorithms and security experiments that can be analyzed in this manner. In Chapter 3, we present the cryptographic primitives required for our blockchain-based steganographic construction.

2.5 Blockchain Overview

The emergence of Bitcoin [52] and other cryptocurrencies has catapulted blockchain to new heights and electrified exploration into a myriad uses and variants [53]. 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. While a complete discussion of blockchain is outside the scope of this paper, we summarize an overview of attributes that we will leverage for creation of a covert channel. We stress that blockchain is chosen as a steganographic channel for practical demonstration due to these unique attributes, but that it is not necessarily the only protocol to feature these attributes.

At its core, blockchain is a simple idea. A block is a chain of blocks, with each block connected to the one before it and after it by means of a mathematical cryptographic relationship. Each block and the events they encode are propagated throughout a distributed network where each node connected to the network can view a record of all events. This concept was introduced by Haber and Stornetta in 1991 in their work regarding computationally practical procedures for the digital time-stamping of documents [54]. Specifically, they proposed the use of one-way functions (typically hashing see [55]) to ensure that a document had not been tampered and that by using a chain of signatories, the provenance of a document could be verified. This idea was extended by Nakamoto to include a series of monetary transactions in each block that would then be codified by this chained hashing system. Figure 2.4 shows a truncated example of this process in the context of Bitcoin.

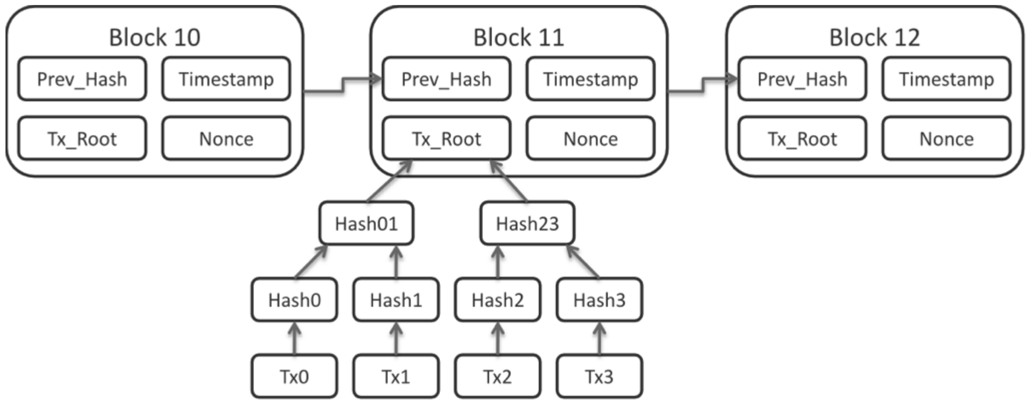

Figure 2.4. Blockchain Example. Source: [56].

While there is confusion about the specific components of blockchains, a blockchain must have three critical elements. These are blocks, events/a record of those events (called transactions and the digital ledger respectively), and a way to agree about the state of the network (consensus). We present the following generic definitions for these structural elements of blockchain.

A block is a container for data. It includes (but is not limited to) the following elements, its index, the hash of the previous block, a timestamp, and a set of transactions. Each block can be identified by its self-identifying hash, which takes as input the elements above. In this way, if an element of a block is changed, the hash of the block is also changed. As the blockchain gets longer, it becomes increasingly computationally difficult for an adversary to modify a block in the past as each subsequent block must also be modified [52].

A transaction is a transfer of data from one party to another. The nature of the data can vary based on the application. Blockchain applications have been proposed for a variety of fields including cryptocurrency [52], [57], medicine [58], supply chain [59], and cyber defense [60], [61]. In the context of cryptocurrency or financial applications, a transaction would codify a transfer of a financial asset from one entity to another. In a medical application, transaction data could contain patient medical records or medical equipment records. In supply chains, the data could contain the production record of a particular product. In a cyber defense setting, a transaction could codify alert data regarding a potential attack. Every transaction is added to a digital ledger that acts as a record or all transactions that have occurred. Each participating node in a blockchain network has to agree about the state of the network (through a process called consensus) and has access to the ledger of transactions at any given time. The flexible nature of this transaction framework makes blockchain a serviceable tool for use in several different applications.

While there are many different types of blockchains [62], [63] (e.g., centralized, decentralized, permissioned, permissionless, public, private, consortium), in this work, we restrict our attention to permissionless blockchains. Permissionless blockchain-based protocols allow any node to join and participate within their networks. Unquestionably, the feature that makes permissionless blockchain-based protocols so attractive for steganographic applications is the immutable public ledger that can be contributed to by any actor. Each data transfer (referred to as transactions) between blockchain participants is recorded in the ledger, which is publicly viewable. Furthermore, these transactions cannot be altered once they are submitted or removed. There is no centralized authority that guarantees the integrity of the transactions; this is instead handled via distributed consensus of the participating nodes in the blockchain network using the specific consensus algorithm.

While blockchain gained attention in the 2000s, the core concept of consensus in distributed networks is not particularly new. Designing systems in which nodes can agree upon the state of a network is a well-researched problem. Consider the following problem: a network of n nodes must agree on the state of a network or particular operation in the presence of f faulty nodes. As pointed out by the seminal work of Lamport, Shostak, and Pease, this situation can be abstractly described by the Byzantine General's problem [64]. They present a scenario in which some number of generals of the Byzantine army are camped around a city. Using only unreliable communications (in this case a messenger), the generals must decide upon a common battle plan. Unfortunately, some number of the generals can be traitors who will attempt to lure the loyal generals into destruction. The challenge is to design a system such that only the loyal generals will reach agreement. It was proven that the problem is solvable only if more than two-thirds of the nodes are loyal [64], although other variants of the problem may provide different bounds (Like Proof-of-Work, which is vulnerable to the 51% attack [65]). To be a reliable communications network of distributed nodes blockchain systems must provide Byzantine Agreement.

There are many examples of consensus algorithms that reach Byzantine Agreement, some of the most well-known being Proof-of-Work (PoW) [52], Proof-of-Stake (PoS) [66], and Practical Byzantine Fault Tolerance (PBFT) [67]. The subject of consensus is particularly important in this work since if the Warden can alter blocks and transactions, they are able to disrupt the underlying steganographic channel used by the prisoners. Thus, the reliability of our steganographic scheme is directly related to the reliability of the blockchain protocol. A detailed comparison of consensus algorithms can be found in [68]. As our work uses Ethereum, we cover its consensus algorithm in Section 2.6.

2.6 Ethereum Internals

We now present an overview of Ethereum internals that will be relevant for our steganographic approach in Chapter 5. We describe blocks, transactions, address generation, and the consensus process in Ethereum.

Ethereum Block Internals A block in Ethereum may include the following elements [69]:

- timestamp: The record of when the block was mined
- blockNumber: The number of the block, also signifies the length of the blockchain
- baseFeePerGas: The minimum fee required for a transaction to be included in the specific block
- difficulty: The mining effort required for block production
- mixHash: The unique block identifier

- parentHash: The unique identifier of the previous block
- transactions: The set of transactions in the block
- stateRoot: The state of the system including account balances, contract storage, contract code, and amount nonces
- nonce: The value that when combined with the mixHash, proves the block matches the consensus algorithm (Proof-of-Work).

According to the Ethereum documentation, the average block time, or the time it takes to mine a new block, is around 12-14 seconds. This time can be managed by adjusting the *difficulty* value for the following blocks. Finally, it is important to note that the maximum size of the blocks is bounded by the amount of gas (i.e., computational work) required to process all of the transactions in the block.

Ethereum Transaction Internals A transaction in Ethereum may include the following fields [69]:

- recipient: The receiving address
- signature: The sender identifier generated when the sender signs the transaction with their private key
- value: The amount of ETH to transfer from sender to recipient
- data: Optional data field
- gasLimit: The maximum amount of gas units that can be consumed by the transaction
- gasPrice: The transaction fee the sender pays per unit of gas

An example of a signed JSON-RPC response for a transaction is shown below.

```
{
 "jsonrpc": "2.0",
 "id": 2,
 "result": {
   "raw": "0xf88380018203339407a565b7ed7d7a678680a4c162885bedbb695
   fe080a44401a6e4000000000000000000000000000000000000000000000000
   000000000000001226a0223a7c9bcf5531c99be5ea7082183816eb20cfe0bbc
   322e97cc5c7f71ab8b20ea02aadee6b34b45bb15bc42d9c09de4a6754e70009
```

```
      08da72d48cc7704971491663",
      "tx": {
        "nonce": "0x0",
        "gasPrice": "0x1234",
        "gas": "0x55555",
        "to": "0x07a565b7ed7d7a678680a4c162885bedbb695fe0",
        "value": "0x1234",
        "input": "0xabcd",
        "v": "0x26",
        "r": "0x223a7c9bcf5531c99be5ea7082183816eb20cfe0bbc322e97cc5c
        7f71ab8b20e",
        "s": "0x2aadee6b34b45bb15bc42d9c09de4a6754e7000908da72d48cc77
        04971491663",
        "hash": "0xeba2df809e7a612a0a0d444ccfa5c839624bdc00dd29e3340d
        46df3870f8a30e"
      }
    }
  }
```

There are multiple protections built into this transaction structure. The hash of the transaction is signed with the sender's private key allowing any participant in the network to verify the integrity of the transaction as well as the identity of the sender. Ethereum uses ECDSA as its signature algorithm. Furthermore, replay protection is provided by use of a nonce (really a sequence number in this case) in the transaction structure.

Ethereum Address Generation In Ethereum, addresses (for both the sender and the receiver) are 160 bits or 40 hexadecimal characters [70]. The process for generating addresses is detailed in [70], [71] and is summarized below and taken from [2].

1. Generate a random private key of 64 hex characters (256 bits).
2. Using ECDSA and the secp256k1 standard, generate the public key (512 bits) with the private key.
3. The account public address is given by the last 20 bytes (160 bits) of the Keccak-256 hash of the public key.

Consensus in Ethereum Currently, Ethereum uses a PoW consensus protocol called Ethash, though it plans to upgrade to a PoS consensus protocol [72]. Ethash was built on Dagger-Hashimoto [73], which was introduced as a secure mining algorithm designed around being efficient and easily verifiable by resource-limited nodes (light client), as well as resistant to application-specific integrated circuit mining (a huge drawback of Bitcoin PoW [74]). The Dagger algorithm makes use of directed acyclic graphs to build large data structures that could achieve memory-hard computation but memory-easy validation. The Hashimoto algorithm achieved ASIC resistance by making memory (RAM) the limiting factor in mining. Thus, this algorithm is IO-bound. Though Dagger-Hashimoto has been superseded by Ethash, many of its components emerge in the following description of the Ethash process (full documentation in [75]):

1. Calculate a seed based on previous block headers.
2. Using the seed, compute a 16 MB pseudorandom cache.
3. Using the cache, generate a large dataset, the DAG that by specification was 1 GB, but is now 4.77 GB [76]. Each item in this dataset should only depend on a small number of items from the cache. This dataset is updated as time passes (every 30000 blocks) and is used for the mining process.
4. The mining process grabs random slices of the dataset and hashes them together. Due to this process, verification can be done with very little memory using the much smaller 16 MB cache to generate the parts of the dataset that were required.

A depiction of the process can be seen in Figure 2.5. This algorithm is GPU friendly and uses the Keccak family of hashing algorithms (Keccak-256 and Keccack-512).

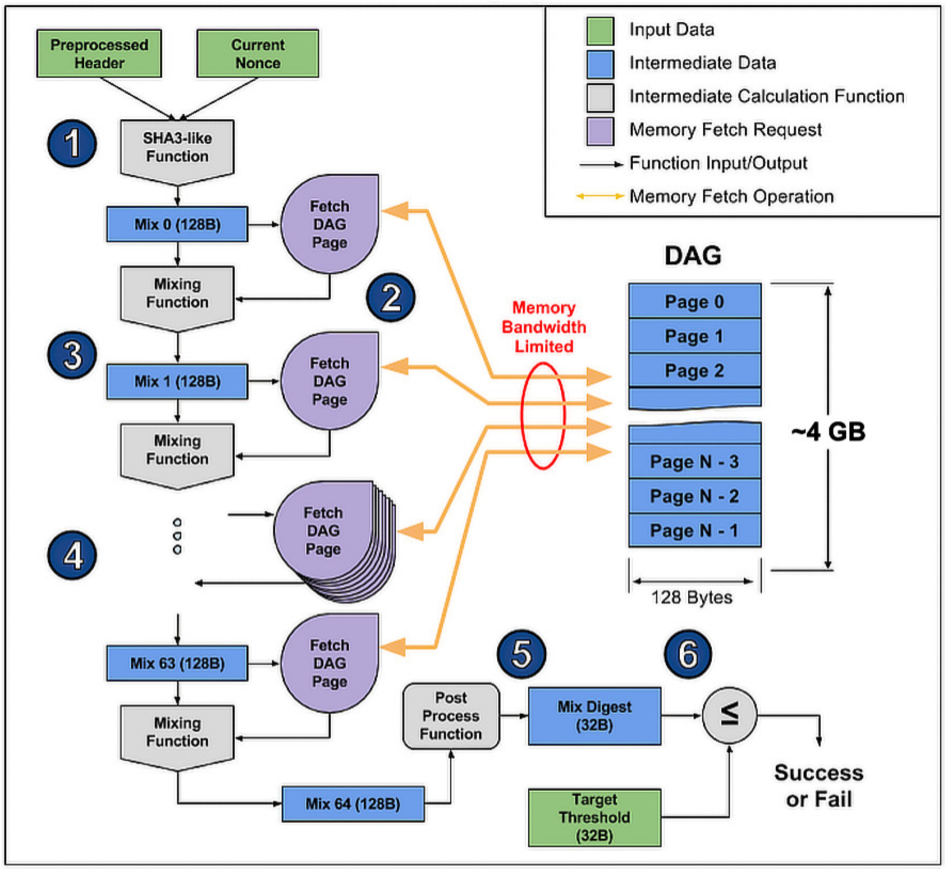

Figure 2.5. Ethash Mining Algorithm. Source: [77].

2.7 Blockchain-based Covert Channels

Finally, we present a number of embedding approaches in blockchain-based protocols. Since their introduction, there have been several efforts to embed information in blockchain-based protocols like Bitcoin and Ethereum. We give special attention to [21] where the authors present *MoneyMorph*, an instantiation of their proposed *stego-bootstrapping* scheme. They present a number of provably secure embedding schemes in Bitcoin, Zcash, Monero, and Ethereum. Their scheme is only valid in blockchain-based protocols where public key cryptography is used. In contrast, though our practical construction proof-of-concept also leverages public key cryptography, our ASAD framework can be used to evaluate other protocols than those that are blockchain-based. We point to specific characteristics of our proof-of-concept scheme 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 [21].

In [20], the author uses an address embedding scheme with least significant bit embedding and analyzes it using the CHA construct [29]. 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.

Bitcoin-based embedding schemes are presented in [78] and [79]. There has also been significant work in covert embedding in the internals of digital signatures. In [80], a modification to ECDSA using a proposed kleptographic algorithm is presented to embed covert data. In [81], a novel, asymmetric embedding scheme using ECDSA is presented. Additionally, in [82], the *value* field in an Ethereum transaction is leveraged to implement an HMAC-based embedding scheme. An address-based embedding scheme in the Ethereum platform is proposed in [2]; the authors use a pseudo-random number generator to select transaction address embedding bit positions. There are many other methods that have been used for blockchain-based embedding, but these examples provide some breadth to this area.

Most of these works use a specific embedding approach or a specific protocol, and security assessment are based on individualized stego-goals. In some cases, the security assumptions are not even well-defined. A major contribution of this paper is presentation of a framework by which steganographic approaches can be generically evaluated. We provide a thorough framework (ASAD) to evaluate stego-embedding security both on and off the blockchain. Furthermore, we propose a practical embedding scheme (ASGARD) using Ethereum transaction address fields as the embedding mechanism, with improved throughput over prior address-based approaches. In fact, our scheme provides the maximum possible throughput (the entire address) for an address-based embedding scheme. The security of ASGARD is analyzed in the ASAD model, achieving the topic level of the security hierarchy.

2.8 Finding a Best-fit Distribution

Much of our work in Chapter 5 and in Chapter 6 revolves around finding a best-fit distribution for activity on our chosen covert communication channel. As such, we present an overview of some of the metrics that will be used in this work as well as a brief summary of some of the distributions that we encountered. Our main tool for determining the best-fit distribution for our data was MATLAB, specifically, the *fitmethis* [83] function in MATLAB and the *distribution fitter* application in MATLAB.

MATLAB uses maximum likelihood estimates (MLEs) to maximize the fit of a chosen distribution to an observed dataset. MLE is used to determine the best estimates for the parameters of a distribution. For example, for a normal distribution, MLE would be used to get the best estimate of the mean μ and the standard deviation σ. We now present a series of definitions to describe the MLE process and related concepts.

First, we define the likelihood function. Given a random sample $X_1, X_2, \cdots, X_N$ from a random variable X and a set of unknown parameters $\theta_1, \theta_2, \cdots, \theta_m$ in a parameter space Ω, the PDF for each X_i (where $i \in N$ and $k \in m$) is $f(x_i; \Omega_k)$. The joint PDF of $X_1, X_2, \cdots, X_N$ or likelihood $L(X, \Omega)$ is [84]:

$$L(X,\Omega) = P(X_1 = x_1, X_2 = x_2, \cdots, X_n = x_n) = f(x_1;\Omega)\cdot f(x_2;\Omega)\cdots f(x_n;\Omega) = \prod_{i=1}^{n} f(x_i;\Omega) \tag{2.8.1}$$

The MLE of X is defined as the parameter values of Ω that maximize Equation 2.8.1 [85]:

$$\hat{\Omega}_{ML} = \text{argmax} \prod_{i=1}^{n} f(x_i;\Omega) \tag{2.8.2}$$

Oftentimes, the natural log is taken of Equation 2.8.2 in order to make the product into a sum as well as to avoid floating point issues. That form is:

$$\hat{\Omega}_{ML} = \text{argmax} \sum_{i=1}^{n} \ln f(x_i;\Omega) \tag{2.8.3}$$

The criteria used to determine the parameter best fit were log-likelihood (LL) and the Akaike information criterion (AIC). Log-likelihood for MLE is defined similarly to Equation 2.8.1 and AIC defined as [86]:

$$\phi_{AIC} = -2\log(\mathcal{L}(\theta;\hat{\Sigma})) + 2K \tag{2.8.4}$$

where $\log(\mathcal{L}(\theta;\hat{\Sigma})$ is the numerical value of the log-likelihood at its maximum point for each parameter θ in Σ and K is the number of estimable parameters in the approximating model. A more detailed description of these criteria and model selection can be found in [86]. As a general rule, if the AIC for one model is more than 2 less than another, that model is superior [87]. We now present an overview of some of the common distributions we encountered in our work.

Log-normal Distribution

The PDF of the log-normal distribution is [88]:

$$f(x \mid \mu, \sigma) = \frac{1}{x\sigma\sqrt{2\pi}} \exp\left\{\frac{-(\log x - \mu)^2}{2\sigma^2}\right\}, \text{for } x > 0 \tag{2.8.5}$$

where μ is the mean and σ is the standard deviation. The CDF of the log-normal distribution is defined as:

$$F(x \mid \mu, \sigma) = \frac{1}{\sigma\sqrt{2\pi}} \int_0^x \frac{1}{t} \exp\left\{\frac{-(\log t - \mu)^2}{2\sigma^2}\right\} dt, \text{for } x > 0 \tag{2.8.6}$$

Figure 2.6 shows how the shape of the log-normal distribution changes as the parameters vary.

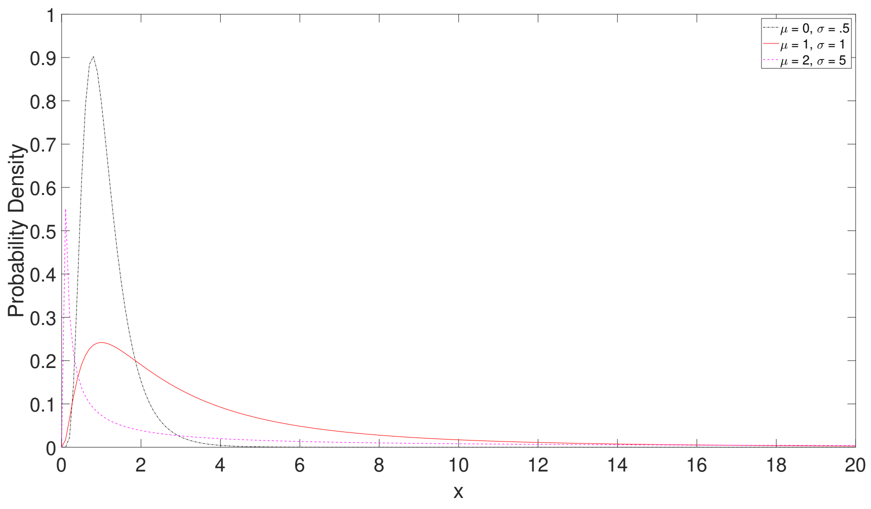

Figure 2.6. Log-normal Distribution with Varying Parameters

The log-normal distribution describes a random variable whose logarithm is normally distributed. It has a variety of applications as many natural phenomena exhibit this trait. Some examples include, economics, finance, reliability analysis, and wireless communications.

Beta Distribution

The PDF of the beta distribution is defined as [89]:

$$f(x \mid a, b) = \frac{1}{B(a, b)} x^{a-1}(1-x)^{b-1} \tag{2.8.7}$$

where a is the first shape parameter, b is the second shape parameter, and $B(\cdot)$ is the Beta function defined as [90]:

$$B(a, b) = \int_0^1 t^{a-1}(1-t)^{b-1}\, dt \tag{2.8.8}$$

The generalized beta distribution also takes two additional parameters, a lower bound and an upper bound. In the standardized beta distribution, these are 0 and 1 respectively. The CDF of the beta distribution is defined as:

$$F(x \mid a, b) = \frac{1}{B(a,b)} \int_0^x t^{a-1}(1-t)^{b-1}\,dt \tag{2.8.9}$$

Figure 2.7 shows how the shape of the beta distribution changes as the parameters vary.

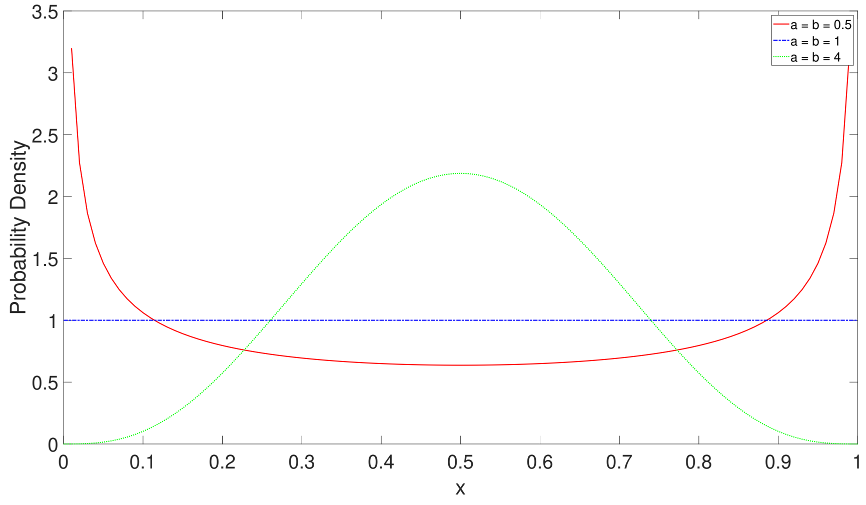

Figure 2.7. Beta Distribution with Varying Parameters

The beta distribution is used to model a variety of use cases including, in order statistics, subjective logic, among others, but is exceptionally useful in the field of machine learning. It serves as a conjugate prior in Bayesian inference for a number of other distributions including, the Bernoulli, binomial, and geometric distributions [91].

Gamma Distribution

The PDF of the gamma distribution is defined as [92]:

$$f(x \mid a, b) = \frac{1}{b^a \Gamma(a)} x^{a-1} e^{\frac{-x}{b}} \tag{2.8.10}$$

where a is the shape parameter, b is the scale parameter, and $\Gamma(\cdot)$ is the Gamma function defined as [93]:

$$\Gamma(a) = \int_0^\infty t^{a-1} e^{-t}\, dt \tag{2.8.11}$$

The CDF of the gamma distribution is defined as:

$$F(x \mid a, b) = \frac{1}{b^a \Gamma(a)} \int_0^x t^{a-1} e^{\frac{-t}{b}}\, dt \tag{2.8.12}$$

Figure 2.8 shows how the shape of the gamma distribution changes as the parameters vary.

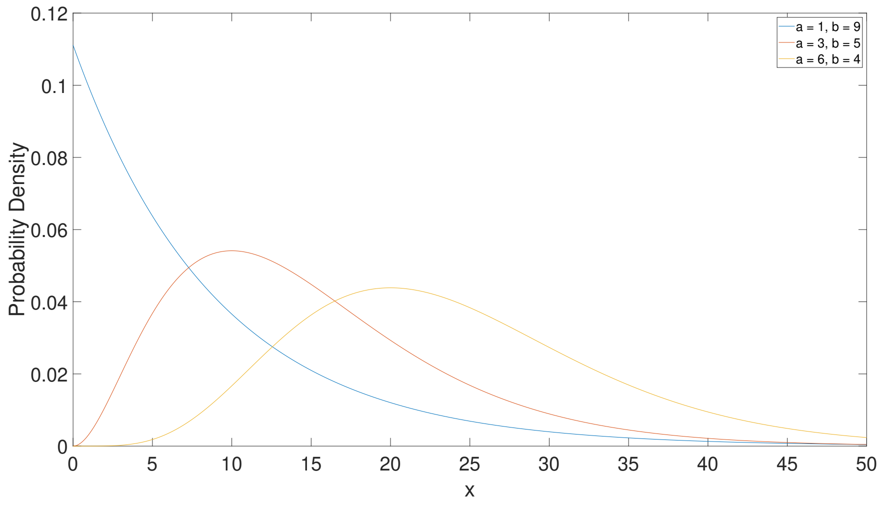

Figure 2.8. Gamma Distribution with Varying Parameters

The gamma distribution models a number of different scenarios including, time to failure for equipment, load levels in telecommunication applications, rainfall, and insurance claims [94]. The key for this distribution is that the data must be positive and skewed. It also serves as the conjugate prior for the Poisson and exponential distributions.

2.9 Summary

In this chapter, we presented important foundational concepts related to steganography, secure channels, blockchain, Ethereum, and blockchain-based covert channels, as well as an overview of distribution fitting, that informed our research. These foundational concepts were critical to the development of the following models, frameworks, algorithms, and embedding approaches.

THIS PAGE INTENTIONALLY LEFT BLANK

CHAPTER 3:
A Steganographic Model Incorporating Anonymity

The purpose of this chapter is to present two different, but related models/frameworks. The first model, the anonymized steganographic model, considers the role of anonymity within the steganographic context. We then present a provable security framework to assess the security characteristics of embedding approaches that fit within our anonymized steganographic model. [7] [8]

3.1 Conceptual Anonymization Framework

The Prisoners' Problem described in Section 2 presupposes that the Warden has the ability to positively identify their prisoners as well as their methods of communication. Privacy-based and anonymity-based applications present an additional layer of complexity for the Warden as they prevent that identification process. This layer of complexity can be used to extend the Prisoners' Problem into what we have called the Anonymized Prisoners' Problem.

3.1.1 Anonymized Prisoners' Problem

A jail contains *n* prisoners with any 2 or more able to conspire to hatch an escape plan. There are authorized opportunities for prisoners to communicate (i.e., yard time) with each other. The Warden is privy to all communications at all times. Unfortunately, the Warden may not be able to positively identify one or more of the communicating prisoners due to the quality of their monitoring equipment (i.e., audio recording quality). We also add two more additional considerations. The first is that the Warden's record of communications is eternal. There are no storage limitations or altering of communications once they are recorded. Furthermore, every escape plan hatched between prisoners has a time limit. The Warden cannot prevent an escape plan that they do not have foreknowledge of. While there

[7]Portions of this chapter were previously published by IEEE [2]. Reprinted, with permission, from V. Kanth, C. Bollmann, M. Tummala, and J. McEachen, "A Novel Adaptable Framework for Covert Communications in Anonymized Protocols", 2021 IEEE Military Communications Conference (MILCOM), December, 2021.

[8]Portions of this chapter will be published by IEEE. Reprinted, with permission, from V. Kanth and B. Hale, "Blockchain-based Authenticated Stego-Channels: A Security Framework and Construction", 2022 IEEE International Conference on Blockchain (Blockchain), August, 2022.

are several other constraints and conditions that can be added to this model, we believe it is sufficiently broad to become a starting point for discussion.

3.1.2 Anonymized Steganographic Model

Clearly, Figure 1.3 is no longer sufficient to describe this steganographic model. Using the conceptual problem outlined by the Anonymized Prisoners' Problem, we incorporate the principle of anonymity into a steganographic model as shown in Figure 3.1 using the idea of anonymity sets from [31]. However, we call these sets identity sets as they describe the sets of pseudonyms/identities that a sender can use rather than the identities themselves. Table 3.1 presents the various elements of the model.

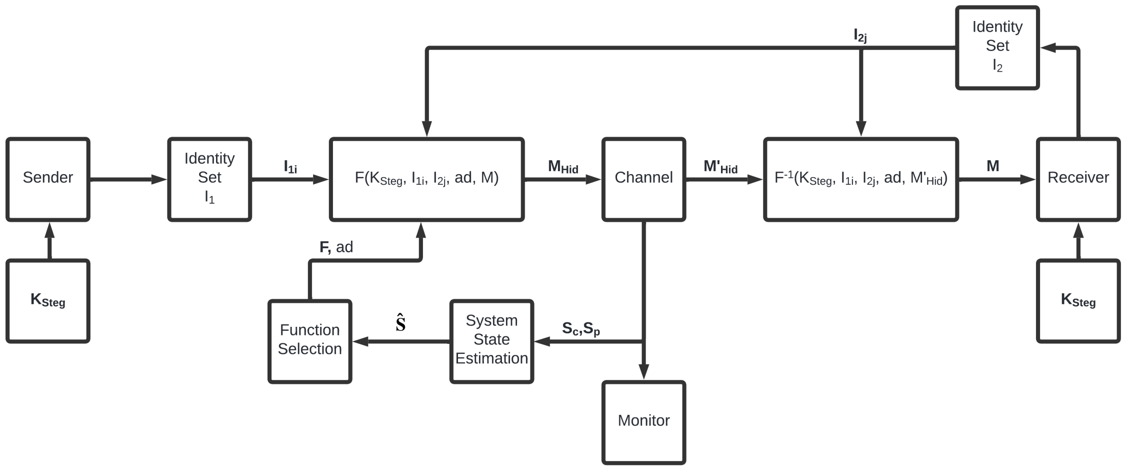

Figure 3.1. Anonymized Secure Communications Model

The Steganographic Key: k_{steg}

While the specifics of key sharing algorithms are beyond the scope of this research, a shared key is necessary for this steganographic model. More generally, a shared key is required for almost all steganographic applications [95]. This key describes where the message M, will be embedded. This is different than a key that would be used to encrypt the message. Also, that encryption operation would be an add-on to our model and is not reflected in Figure 3.1. Our constructions delineate keys for different operations as required. In this dissertation, we assume that the relevant key exchanges have already taken place.

Table 3.1. Elements of the Anonymized Steganographic Model

Element	Description
k_{steg}	The embedding key shared between the sender and receiver
I_1 and I_2	The identity sets for the sender and receiver
F and F^{-1}	The steganographic function and its inverse
S_c	The set of channel state observations
S_p	The channel protocol
$\hat{S}$	The estimation of the channel state
M, M_{hid}, M'_{hid}	The desired message, hidden message, and hidden message after channel traversal
ad	The associated data sent alongside the stegotext (i.e., the embedded message)

Identity Sets

The sets I_1 and I_2, are the collections of identities that the sender and receiver can assume for the purposes of communication. These sets are limited both by protocol and by the needs of secure communication methodologies. In the abstract, any anonymized protocol will use some string for entity identification purposes. While that string may or may not be able to be tied to an actual identity, it is necessary for any data transfer to take place. Transactions must have a sender and a receiver. The number of identities that an entity can take may be limited by a particular protocol, but many services like Bitcoin and OpenSig (a digital signature technology) actually recommend using several different identities [96] [97].

In some cases, the defining of identity sets can be quite straightforward. For example, in the case of Ethereum the set of identities that the sender and receiver can assume is defined by the Ethereum public address. This address is derived by taking the last 20 bytes of the hash of the public key. Each bit of the 20 bytes has a $\frac{1}{2}$ probability of being either 0 or 1. Thus the total number of identities that could be used in Ethereum would be 2^{160}. Interestingly, Ethereum allows transactions to be sent to oneself so the max identity set is the same for both the sender and receiver.

The Steganographic Functions: F and F^{-1}

The functions F and F^{-1} correspond to the steganographic functions utilized in the secure communications scheme. They take as input the shared key k_{steg}, the specific identities of the sender I_{1i} and receiver I_{2j}, and the desired message M. The function is selected based on the sender's estimation of the channel state ($\hat{S}$) that uses channel state observations (S_c) and the channel protocol (S_p) to drive the estimation process.

Channel State Estimation: $\hat{S}$

The state of the communication channel can play an important role in the steganographic process. For example, we may only want to send a message at a time where the message traffic rate is high as to minimize detectability. In the case of cryptocurrency, we may want to target times of low traffic as transaction costs will be lower. Depending on the application protocol, use case, and desires of the communicating entities, an accurate understanding of the channel state is crucial. Thus, this model includes a set of channel state observations (S_c) that capture various aspects of channel activity at a particular time t. Those observations, in conjunction with the channel state protocol (S_p), are the inputs to the channel state estimation block.

The Messages: M, M_{hid}, M'_{hid}

The message M, contains the information that Alice is attempting to send to Bob. It is important to note that the message does not contain the associated data ad, rather, it is grouped with the associated data and sent across the channel as part of M_{hid}. From a vocabulary perspective, once the message has been embedded in the specified location/field, that data element is called the stegotext.

The Associated Data: ad

Associated data is data that must be sent alongside our message in order to constitute a legitimate protocol transaction. For example, in the case of IP-based communication, the associated data might be packet headers. In the context of least-significant bit (LSB) embedding in a 2D image, the associated data would be all of the other seven most significant bits in each pixel. Figure 3.2 shows what elements would comprise the associated data in this case. In the context of our steganographic framework, the associated data is all of the other fields outside the field that contains our message M.

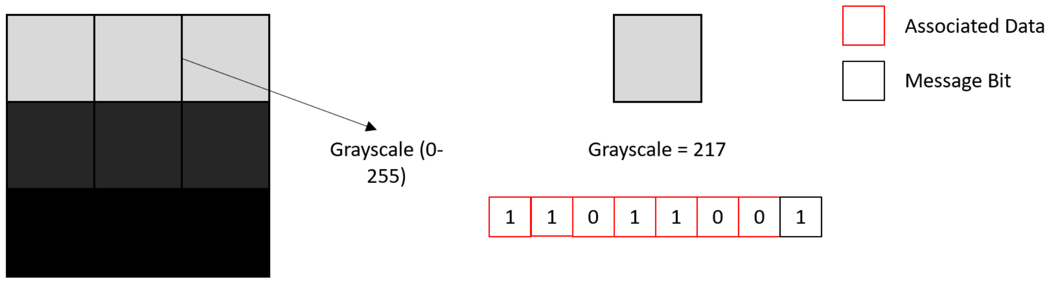

Figure 3.2. Associated Data in an Image LSB Embedding Scheme. Adapted from: [98].

The Messages: M$_{hid}$ and M'$_{hid}$

The messages M$_{hid}$ and M'$_{hid}$ are the outputs of their respective steganographic functions. They contain the message M and its associated data ad. Based on the type of channel, either noisy or noiseless, M$_{hid}$ and M'$_{hid}$ could be different and may require an additional layer of error correction to read correctly. In our case, our blockchain-based embedding construction and schemes are built on application layer protocols and can be treated as noiseless. Figure 3.3 shows an example of M$_{hid}$ given an arbitrary protocol with n fields, and the message embedded in field 2. We highlight again that the embedded message is called the stegotext. Note that M may or may not be encrypted depending on the steganographic function and the underlying covertext field.

3.2 Cryptographic Primitives

This section contains the relevant cryptographic primitives and their security properties as relevant to the creation of steganographic model and our embedding scheme.

3.2.1 Symmetric Encryption

Borrowing from [51] and [49] we present the following formulation for a symmetric encryption scheme.

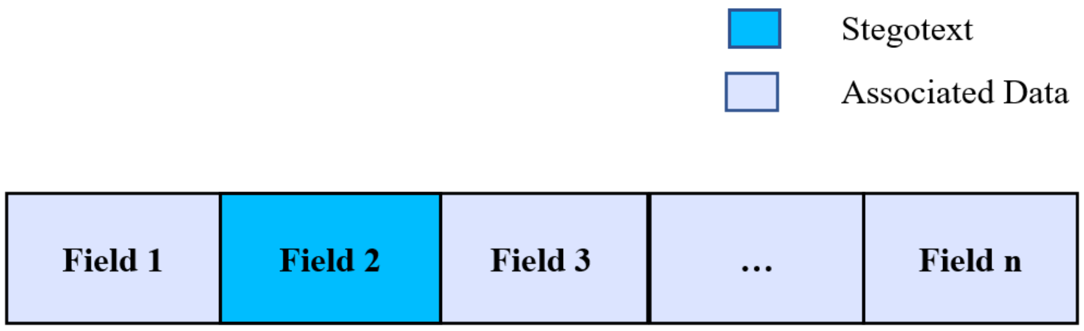

Figure 3.3. Example of M_{hid}

***Definition* 3.2.1.** *A symmetric encryption scheme defined as* $\Pi = (\mathrm{Kgn}, \mathrm{Enc}, \mathrm{Dec})$ *consists of three algorithms where* $\mathcal{K}$ *is the key space,* $\mathcal{M}$ *is the message space, and* $\mathcal{C}$ *is the ciphertext space:*

- $\mathrm{Kgn}(1^\lambda) \xrightarrow{\$} k$*: A probabilistic key generation algorithm that takes as input* 1^λ *where* λ *is the security parameter. The function outputs a key,* $k \in \mathcal{K}$.
- $\mathrm{Enc}(k, m) \xrightarrow{\$} c$*: An algorithm that takes as input a key* $k \in \mathcal{K}$ *and a plaintext* $m \in \mathcal{M}$ *and outputs a ciphertext* $c \in \mathcal{C}$.
- $\mathrm{Dec}(k, c) \xrightarrow{\$} m$*: An algorithm that takes as input a key* $k \in \mathcal{K}$ *and a ciphertext* $c \in \mathcal{C}$ *and outputs a plaintext* $m \in \mathcal{M}$.

An encryption scheme Π satisfies correctness if for all $k \in \mathcal{K}$ and for all $m \in \mathcal{M}$,

$$\Pr\Big[\mathrm{Dec}(k, \mathrm{Enc}(k, m)) = m\Big] = 1.$$

Indistinguishability Under Chosen-Plaintext Attack (IND-CPA)

The security notion that our algorithm leveraging symmetric encryption requires is indistinguishability under chosen-plaintext attack. In this attack model, the probabilistic polynomial time (PPT) adversary has access to an encryption oracle. The following IND-CPA experiment is presented as in [99]:

1. A key k is generated by running $\mathrm{Kgn}(1^\lambda)$.
2. The adversary makes a sequence of calls to the encryption oracle for arbitrary plaintext.
3. The adversary submits two messages $m_0, m_1 \in \mathcal{M}$ where $|m_0| = |m_1|$. The encryption oracle responds by sampling $b \xleftarrow{\$} \{0,1\}$ and computing $c^* \xleftarrow{\$} \mathrm{Enc}(k, m_b)$; it returns the ciphertext c^*.
4. The adversary can continue to make calls to the oracle.
5. The adversary outputs $b' \in \{0,1\}$.

The advantage of $\mathcal{A}$, denoted $\mathbf{Adv}^{\textsf{IND-CPA}}_{\Pi,\mathcal{A}}(\lambda)$, is equal to $\left|\Pr[b = b'] - \frac{1}{2}\right|$ in the above experiment.

***Definition* 3.2.2.** *A symmetric encryption scheme* $\Pi = (\mathrm{Kgn}, \mathrm{Enc}, \mathrm{Dec})$ *is secure against Chosen Plaintext Attacks if, for all PPT adversaries $\mathcal{A}$, there is a negligible function* negl *such that*

$$\mathbf{Adv}^{\textsf{IND-CPA}}_{\Pi,\mathcal{A}}(\lambda) \leq \mathrm{negl}(\lambda)\ .$$

3.2.2 Hashing

A hash function takes a variable length input and returns a fixed length output. This can be described in the following manner:

***Definition* 3.2.3.** *A hash function with output length n is a polynomial-time algorithm H satisfying the following*

- $\mathrm{H}(x) \xrightarrow{\$} t \in \{0,1\}^n$*: H takes as input a string $x \in \{0,1\}^*$, and outputs a string t of length n.*

Security experiments for hash functions range across collision resistance, preimage resistance, and second preimage resistance. In this work, we rely on the pseudorandomness of H.

3.2.3 Digital Signature

For the work in the following sections, we furthermore rely on digital signatures, and borrow the definition of [100]:

***Definition* 3.2.4.** *A digital signature scheme consists of three PPT algorithms* (Kgen, Sign, Vfy) *such that:*

- $\mathrm{Kgn}(1^\lambda) \xrightarrow{\$} (pk, sk)$*: A probabilistic key generation algorithm that takes as input* 1^λ *where* λ *is the security parameter, and outputs a pair of public and private keys,* (pk, sk)*.*
- $\mathrm{Sign}(sk, m) \xrightarrow{\$} \sigma$*: A potentially probabilistic algorithm* Sign *takes as input a private key* sk *and a message* $m \in \{0, 1\}^*$*, and outputs a signature* σ*.*
- $\mathrm{Vfy}(pk, m, \sigma) \xrightarrow{\$} b \in \{0, 1\}$*: A deterministic verification algorithm* Vfy *takes as input a public key* pk*, a message* $m \in \{0, 1\}^*$*, and a signature* σ*, and outputs a bit* $b \in \{0, 1\}$*. If* $b = 0$*, the signature is considered invalid and if* $b = 1$*, the signature is considered valid.*

A digital scheme satisfies correctness if, for all pairs (pk, sk) output by Kgn and for all messages $m \in \{0, 1\}^*$,

$$\Pr\Big[\mathrm{Vfy}(pk, m, \mathrm{Sign}(sk, m)) = 1\Big] = 1.$$

We abuse notation and consider equivalency in $\mathrm{Vfy}(pk, m, \sigma) = \mathrm{Vfy}_{pk}(m, \sigma)$ and $\mathrm{Sign}(sk, m) = \mathrm{Sign}_{sk}(m)$.

Strong Unforgeability

We require Strong Unforgeability under an Adaptive Chosen-Message Attack. Let Π = (Kgn, Sign, Vfy) be a signature scheme, $\mathcal{A}$ be an adversary, and λ be a security parameter. The signature experiment $\mathrm{Exp}^{\mathsf{SUF\text{-}CMA}}_{\mathcal{A},\Pi}(\lambda)$ is defined in the following manner:

1. Keys (pk, sk) are generated by running $\mathrm{Kgn}(1^\lambda)$.
2. The adversary $\mathcal{A}$ is given the public key pk and access to an oracle $\mathrm{Sign}_{sk}(\cdot)$ allowing $\mathcal{A}$ to output valid message-signature pairs (m, σ). Let Q denote the set of all (m, σ) such pairs.
3. The adversary $\mathcal{A}$ succeeds and the experiment outputs 1 if and only if $\mathrm{Vfy}_{pk}(m, \sigma) = 1$ and $(m, \sigma) \notin Q$.

***Definition* 3.2.5.** *A signature scheme* $\Pi = (\mathrm{Kgn}, \mathrm{Sign}, \mathrm{Vfy})$ *is Strongly Unforgeable under an Adaptive Chosen-Message Attack if for all PPT adversaries* $\mathcal{A}$ *there is a negligible function* negl *such that*

$$\Pr\left[\mathbf{Adv}_{\Pi,\mathcal{A}}^{\mathsf{SUF\text{-}CMA}}(\lambda) = 1\right] \leq \mathrm{negl}(\lambda).$$

3.3 A Steganographic Model Incorporating Associated Data

As introduced in Chapter 2, the steganographic game is played between a sender, receiver, and a warden (monitor) [19]. 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 [41] and its extension [29]. 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 [101] 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.

Look-Ahead Model [41]

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 C_h be the channel distribution conditioned on h. In [41], 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 $C_{h_1 \circ \ldots \circ h_n}$ based on $h = h_1, h_2, \ldots, h_n$ as incremented. We follow the extension of [101], 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 $h = h_n, h_{n+1}, \ldots, h_l$.

Embedding Scheme

An embedding scheme consists of a tuple of algorithms defined in the following manner building off the definition in [102]. As noted in Chapter 2, we extend this to cover the presence of associated data in our embedding scheme.

***Definition* 3.3.1.** *An embedding scheme defined as* $\Pi = (\mathrm{Kgn}, \mathrm{Embed}_{\mathcal{S}}, \mathrm{Decode}_{\mathcal{S}})$ *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:*

- $\mathrm{Kgn}(1^\lambda) \xrightarrow{\$} k_{steg}$: *A possibly probabilistic key generation algorithm that takes as input* 1^λ *where* λ *is the security parameter, and outputs a key,* $k_{steg} \in \mathcal{K}$.
- $\mathrm{Embed}_{\mathcal{S}}(k_{steg}, \mathtt{ad}, m, h) \xrightarrow{\$} s$: *A possibly probabilistic algorithm that takes as input a key* $k_{steg} \in \mathcal{K}$*, associated data* $\mathtt{ad} \in \mathcal{AD}$*, a message* $m \in \mathcal{M}$*, a channel history* h *and outputs a stegotext* $s \in \mathcal{S}$.
- $\mathrm{Decode}_{\mathcal{S}}(k_{steg}, \mathtt{ad}, s) \rightarrow (\mathtt{ad}, m, \alpha)$: *An algorithm that takes as input a key* $k_{steg} \in \mathcal{K}$*, associated data* $\mathtt{ad} \in \mathcal{AD}$*, and a stegotext* $s \in \mathcal{S}$*, and outputs a tuple* $(\mathtt{ad}, m, \alpha = 1)$ *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 been to use the same channel history.*

An embedding scheme Π satisfies correctness except with negligible probability if, for all

$k_{steg} \in \mathcal{K}$, for all $\mathsf{ad} \in \mathcal{AD}$, for all $m \in \mathcal{M}$, and for all channel histories,

$$\Pr\left[\mathrm{Decode}_{\mathcal{S}}(k_{steg}, \mathsf{ad}, \mathrm{Embed}_{\mathcal{S}}(k_{steg}, \mathsf{ad}, m, h)) = m\right] = 1 - \mathrm{negl}(\lambda)\ .$$

Unlike with encryption that requires an absolute decryption for correctness, stego channels account for channel noise and therefore correctness requires a bound of $1 - \mathrm{negl}(\lambda)$ (see [41] [102] for more details).

3.3.1 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 [102] called security against Chosen Hiddentext Attacks ($\mathsf{IND\text{-}CHA}$), which is based on symmetric encryption definitions like those from [51]. Let $\Pi = (\mathrm{Kgn}, \mathrm{Embed}_{\mathcal{S}}, \mathrm{Decode}_{\mathcal{S}})$, λ be the security parameter, and $\mathcal{S}$ be the covertext channel distribution.

***Definition* 3.3.2.** *We say that a steganographic scheme* Π *provides Indistinguishability against Chosen Hiddentext Attacks if for all PPT adversaries* $\mathcal{A}$*, and keys* $k_{steg} \xrightarrow{\$} \mathrm{Kgn}(1^{\lambda})$ *if:*

$$\Pr\left[\mathcal{A}^{\mathsf{Embed}(k_{steg},\cdot,\cdot,\cdot)} = 1\right] - \Pr\left[\mathcal{A}^{\mathsf{S}(k_{steg},\cdot,\cdot)} = 1\right] \leq \mathrm{negl}(\lambda)\ .$$

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

3.3.2 Authenticated Stegotext Associated Data (ASAD) Security

As noted in Chapter 2, 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 receiver is guaranteed the authenticity of the data received.

Many authentication and 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 [27]. 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). The original security definitions from [27] can be found in Appendix A.

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 e.g., IND-CPA security could strengthen the requirements on channel security.

***Definition* 3.3.3.** *Let* Π *be an embedding scheme, let* $\mathcal{A}$ *be a PPT adversary algorithm, 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* $\mathrm{Exp}_{\Pi,\mathcal{A}}^{\mathsf{asad}_i-b}$ *in Figure A.1. We define*

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

Adversary Win Conditions

We present the following winning conditions for the adversary adapted from [27].

- cond_1: To capture basic authentication of stegotexts, we say the adversary has won if the received stegotext s was not one of the stegotexts output by the Embed oracle or the authenticated associated data ad was not the associated data output by the Embed

oracle.

- cond_2: To capture basic authentication of stegotexts with no replays, we say the adversary has won if the received stegotext s was not one of the stegotexts output by the Embed oracle, or if it was output by the Embed oracle and previously accepted by the Decode oracle, or the authenticated associated data ad was not the associated data output by the Embed oracle.
- cond_3: To capture basic authentication of stegotexts with no replays and strictly increasing receipt, we say the adversary has won if the received stegotext s was not one of the stegotexts output by the Embed oracle, or if there were two messages received out of order: if $rcvd_w$ was received before $rcvd_v$, but $rcvd_w$ was sent after $rcvd_v$, or the associated data ad was not the authenticated associated data output by the Embed oracle.
- cond_4: To capture basic authentication of stegotexts with no replays, strictly increasing receipt, and no drops, we say that the adversary has won if the v-th stegotext received (s) is not the v-th stegotext sent, or if fewer than v stegotexts have been sent, or the associated data ad was not the v-th associated data element sent. This captures the idea that the adversary can win if it can succeed in having further messages correctly received.

Relationship between Authenticated Steganographic Notions

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 [27].

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 [27].

Proof. $\mathcal{A}$ starts Π

- *For i = 1.*

As a level-1 adversary, $\mathcal{A}$ wins with a stegotext s_{win}, where $(\nexists w : s_{win} = s_w)$ or an associated data element $\mathtt{ad}_{win}$, where $(\nexists w : \mathtt{ad}_{win} = \mathtt{ad}_w)$. It follows that $\mathcal{A}$ will also win against Π as a level-2 adversary.

- *For i = 2.*
  Case 1: $\mathcal{A}$ wins by forging s_{win} or $\mathtt{ad}_{win}$. This is the same as $i = 1$.
  Case 2: As a level-2 adversary, $\mathcal{A}$ wins with $s_{win} = rcvd_v$, where $\exists w < v$ such that $s_{win} = rcvd_w$ for some $s_u \in s$.
  This is an example of a replay with $s_{win} = rcvd_v = s_y$ for some y, as s_{win} is not a forgery. Let $rcvd_w = s_x$, for some x. If this is the case, $sent_y = rcvd_v = s_{win} = rcvd_w = s_x$. Furthermore, as $\mathcal{A}$ won with a replay, $x = y$. Thus, $\exists x, y, z$ such that $(w < v) \wedge (s_x = rcvd_w) \wedge (s_y = rcvd_v) \wedge (x \geq y)$ ensuring that $\mathcal{A}$ wins as a level-3 adversary.
- *For i = 3.*
  Case 1: $\mathcal{A}$ wins by forging s_{win} or $\mathtt{ad}_{win}$. This is the same as $i = 1$.
  Case 2: As a level-3 adversary, $\mathcal{A}$ wins with $s_{win} = rcvd_v$ where $\exists x, y, z$ such that $(w < v) \wedge (s_x = rcvd_w) \wedge (s_y = rcvd_v) \wedge (x \geq y)$. This implies that $s_w in \neq s_v$. Thus, $\mathcal{A}$ also wins as a level-4 adversary.

$\square$

3.4 Summary

In this chapter, we presented two important models/frameworks. The first, the anonymized steganographic model, extended the' Prisoners' Problem to take into account scenarios in which the Warden cannot positively identify two communicating parties. We then presented a framework to assess the security properties of a generic steganographic scheme. These properties went beyond whether or not the Warden could simply detect a hidden communique, but also considered their ability to potentially force drops, replays, reorderings, or forgeries of valid messages. The combination of the anonymized steganographic model and our provable security framework provide the basis for our analysis of embedding strategies using specific protocols like blockchain-based protocols that purport to provide some semblance of anonymity.

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

```
1: $k_{steg} \xleftarrow{\$} \mathrm{Kgn}()$
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 $(b = b')$
```

Oracle Embed(ad, m, h)

```
1: $u \leftarrow u + 1$
2: $\mathcal{S} \leftarrow \mathcal{C}_h$
3: $sent.s^{(0)} \xleftarrow{\$} \mathcal{S}(k, \mathsf{ad}, h)$
4: $sent.s^{(1)} \leftarrow$
   $\mathrm{Embed}_{\mathcal{S}}(k_{steg}, \mathsf{ad}, m, h)$
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: $(sent.ad_u, sent.s_u) := (\mathsf{ad}, sent.s^{(b)})$
12: return $sent.s_u$ to $\mathcal{A}$
```

Oracle Decode(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: $(\mathsf{ad}, m, \alpha) \leftarrow \mathrm{Decode}_{\mathcal{S}}(k_{steg}, \mathsf{ad}, s)$
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 $\mathcal{A}$
```

Authentication Conditions

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

Figure 3.4. Authenticated Stegotext with Associated Data (ASAD) Experiment, asad_i, with Authentication Condition cond_i for Embedding Scheme $\Pi = (\mathrm{Kgn}, \mathrm{Embed}_{\mathcal{S}}, \mathrm{Decode}_{\mathcal{S}})$ and Adversary $\mathcal{A}$

THIS PAGE INTENTIONALLY LEFT BLANK

CHAPTER 4:
A Blockchain-based Embedding Construction

This chapter introduces a blockchain-based embedding construction, called Authenticated SteGotext with Associated tRansaction Data (ASGARD), that fits within our anonymized steganographic model. We then analyze that model using our ASAD framework and provide an argument for correctness. This chapter serves as a bridge between our theoretical model in Chapter 3 and our Ethereum-based embedding scheme in Chapter 5. [9]

4.1 Blockchain in the Anonymized Steganographic Model with AD

It is worth noting that the inspiration for our anonymized model and our steganographic extension to AEAD was the emergence of blockchain protocols. The wave of new permissionless blockchain technologies and protocols changes the covert communications paradigm. Chapter 2 discusses some of the fundamental attributes of blockchain and permissionless blockchain. A combination of these attributes makes blockchain systems particularly useful for covert communications (though with some drawbacks). The existence of a distributed ledger and consensus theory provides for both data redundancy and easy data access. The validity of the transaction is also assured due to the use of a consensus mechanism and a variety of signature schemes. A permissionless blockchain system allows every participating node to view the transaction ledger in order to achieve consensus. This access to the transaction ledger not only allows nodes to hide messages in valid communications, but also allows all of the parties to make determinations regarding channel state and meets the constraints set by the Anonymized Prisoners' Problem. Furthermore, this ledger transparency actually adds a wrinkle to the formulation. All parties including the prisoners and the Warden have access to all communications data unlike the original presentation of the problem in which only the Warden had that capability. We believe this attribute provides advantages to the scheming prisoners and can be evaluated using our ASAD framework. Figure 4.1 depicts

[9]Portions of this chapter will be published by IEEE. Reprinted, with permission, from V. Kanth and B. Hale, "Blockchain-based Authenticated Stego-Channels: A Security Framework and Construction", 2022 IEEE International Conference on Blockchain (Blockchain), August, 2022.

how blockchain would be used in the context of the Prisoners' Problem with blockchain transactions acting as the vehicle for covert embedding and the blockchain itself acting as the channel.

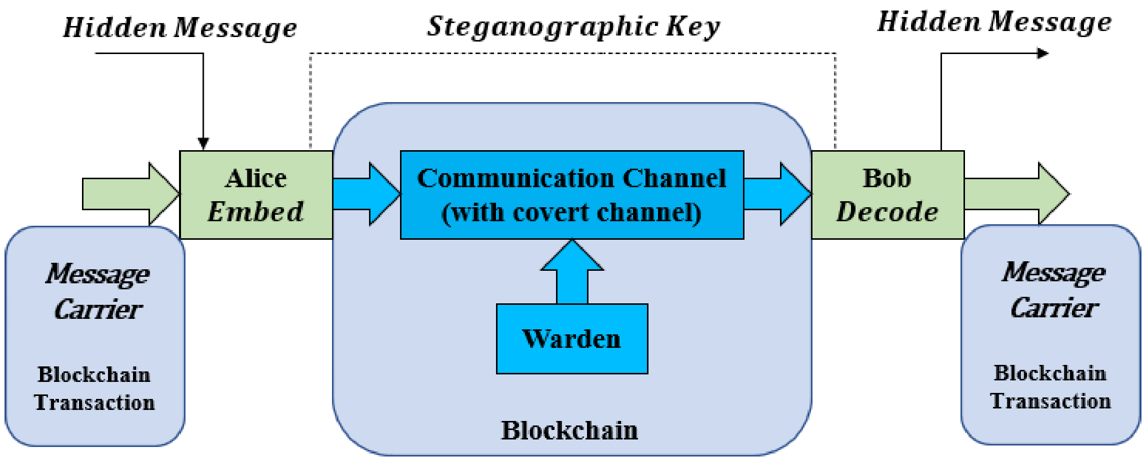

Figure 4.1. Model for Hidden Communication using a Blockchain-based Transaction Carrier. Adapted from: [17].

Anonymity being built into permissionless blockchain protocols is perhaps the biggest advantage of using blockchain covert channels. In the Warden scenario, the Warden has knowledge of both the sender and receiver as they are the prisoners. Figure 3.1 presents the scenario where Alice and Bob can take on one of several identities. Permissionless blockchain protocols like Bitcoin and Ethereum provide precisely this functionality. One can simply change addresses for the sender or receiver to throw off the monitor. What makes this even more difficult for the Warden is that this type of identity switching is not only possible, but generally recommended. Therefore, unlike with many other covert channels, covert communications in blockchain do not rely on manipulating the structure of the protocol. In some sense, permissionless blockchain provides the tools for covert communications in the course of normal operations. Furthermore, the digital ledger allows Bob to access transactions without the Warden's knowledge. We now present a blockchain-based construction that takes advantage of these tools for an ideal blockchain environment.

4.2 ASGARD 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 environment that serves as the platform for data transfer. We then describe variables, the embedding algorithm, and the decoding algorithm.

4.2.1 Ideal Blockchain

We build off of an ideal blockchain-based environment (similar to [20]) 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. More discussion on the Ethereum transaction structure and address generation can be found in Chapter 2. Here, the key components of the structure that are essential to our construction are as follows.

- Entity identities
- Entity addresses (public addresses)
- Transactions
- Channel Ledger

Entity Identities and Addresses The embedding construction takes advantage of the hash functions that are routinely used in blockchain public address generation [70]. 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 that 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 [70] (see Chapter 2 for more details).

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

Figure 4.2 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* [103].

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.

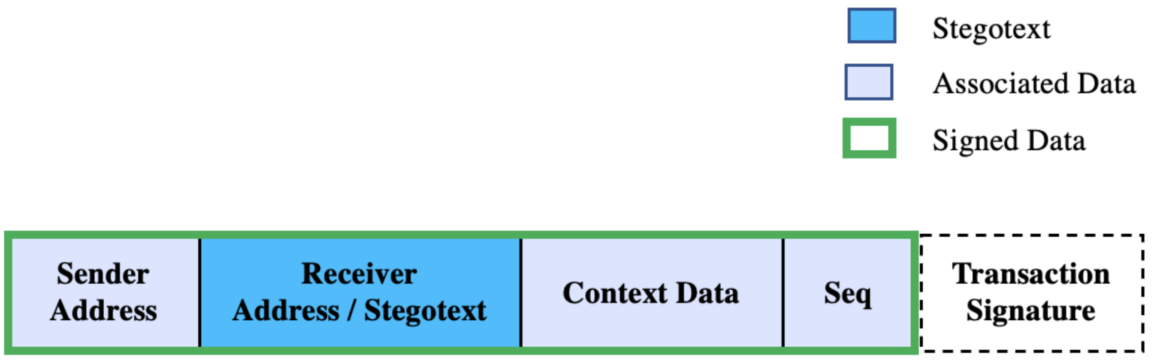

Figure 4.2. Proposed Blockchain-based Transaction Structure with Stegotext

In addition to the field used for stegotext transmission, there are notably several fields of associated data (ad). Associated data is any other set of values required for the transaction. For example, in the context of a cryptocurrency, the ad might include the amount of money being sent from the sender to the receiver. In Figure 4.2, it can be seen that the sender address, context data, and sequence number comprise the ad, all of which is authenticated in addition to the stegotext field re-purposed 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 [70]. 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. In addition, contract accounts start at one. Furthermore, the sequence number length is 64-bits [104]. The Ethereum Yellow Paper [71], as well as an Ethereum Improvement Proposal (EIP) [105] detail these distinctions in more detail.

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 Chapter 2, the state of the network documented in `ledger` is maintained via the Byzantine agreement consensus mechanism associated with the specific blockchain protocol.

4.2.2 Definitions

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

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
- $addr_I :=$ 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 $I = ((pk_I, sk_I), addr_I)$
- `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. 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 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. An identity I corresponds to the public key and address for a unique account.

Remark 3. *We note here that sender addresses should only be used once as address reuse can impact the privacy of communications [106]. This is particularly important in a steganographic context with regards to unlinkability since if Alice repeatedly uses the same sender address, the Warden will potentially be able to positively identify all transactions that Alice is a part of, which could leak information. Using a sender address more than once is not a guarantee of vulnerability, but we avoid it as a matter of good practice. An analysis of potential risks of address reuse is presented in Chapter 6.*

Another important aspect is the role of k_{steg}, the stegotext embedding key described in Chapter 3. For this construction, the embedding key denotes the transaction structure and placement of embedded information. 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 C_h distribution.

Two additional primitives are used in our construction: a symmetric encryption algorithm Enc and a digital signature algorithm Sign. As an intuition, we encrypt the 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 for consensus. Rather it is added to various blockchain-based protocols, such as Ethereum, for a variety of purposes including transaction verification and authentication. Thus, the construction reliance on inclusion of Sign does not interfere with the modular composition with a blockchain public ledger utilizing Byzantine agreement.

4.2.3 Embed Function

What follows is an informal description of our generic address embedding algorithm. Algorithm 1 provides a formal definition of steps 2-4. For our padding process in step 1, a regular block cipher padding algorithm will suffice.

The embed function takes as input k_{steg} (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). The associated data constitutes all auxiliary data.

1. Depending on the length of M and defining l = |address field|
   (a) if |M| = l move to Step 2.
   (b) if |M| $<$ l, pad M until $|M| = |l|$ and move to Step 2.
   (c) if |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.

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 $\texttt{ad} = (addr_I, context, seq)$.

Algorithm 1 provides a more technical description of address embedding given a message M where $|\text{M}| = l$. In the case where $|\text{M}| > l$, we simply run the embedding algorithm for each substring m of M, where $|m| = l$ and pad the last substring as needed (this would happen in step 1.a). 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
    input : $(ID_{priv}, k, \texttt{ledger}), context, M$
    output $(trans, \sigma)$
    :
    $M_e \leftarrow \text{Enc}_k(M)$;
    $((sk_I, pk_I), addr_I) \xleftarrow{\$} ID_{priv}$;
    $IDpriv \leftarrow IDpriv \setminus \{I\}$;
    $trans.sending\_addr \leftarrow addr_I$;
    $trans.recv\_addr \leftarrow M_e$;
    $trans \leftarrow (trans.sending\_addr, trans.recv\_addr, context,$
      $seq)$;
    $\sigma \leftarrow \text{Sign}_{sk_I}(trans)$;
    $\texttt{ad} \leftarrow (trans.sending\_addr, context, seq)$;
    $seq{+}{+}$;
    $\texttt{ledger} \leftarrow \texttt{ledger} \cup \{(trans, \sigma)\}$;
    return ledger;
```

Decode Function

Decoding is split into two parts, an extract function, which is captured by step 1, and a decode function, which is captured by steps 2-5. Our extraction and decode function algorithms are described informally as follows and formally in Algorithms 2 and 3:

1. Monitor the channel record, `ledger` for new transactions
2. If a transaction matches a sender address in ID, move to Step 3. Otherwise return to Step 1.
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 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. Therefore, it functions on a particular channel message (whether stegotext or regular covertext), and thus, individual messages must be first extracted from the ledger-channel, which contains all messages in the environment. Nonetheless, for completeness, we enforce Decode$_{\mathcal{S}}$ to correctly handle messages once received, including checking valid sourcing.

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

```
Decode
  inputs : $(ID_{pub}, k)$, ad, $(s, \sigma)$
  output : (ad, $M$, $\alpha$)
  $(addr_I, context, seq) \leftarrow$ 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 (ad, $M$, $\alpha$);
```

4.2.4 Correctness

Correctness follows directly from the properties of the channel record, `ledger`, and from the properties of the embedding scheme. A message will be extracted from the channel, partitioned by appropriate fields, verified, and the stegotext decrypted using the pre-shared key. In the case where multiple transactions are required to transmit the entire message, the sender can enforce ordering by waiting until the previous transaction has been posted in the channel record `ledger`.

4.3 Security Analysis

In this section, we present a security analysis of the embedding scheme under the look-ahead model presented in Chapter 3. It is worth re-describing the set of conditions that describe an adversarial win. Informal conditions are as follows:

1. The adversary is able to identify a transaction with embedded data from a transaction without embedded data.

2. The adversary is able to modify a transaction initiated by a specific sender address undetected.
3. The adversary is able to replay a transaction initiated by a specific sender address.
4. The adversary is able to reorder transactions initiated by a specific sender address.
5. The adversary is able to drop transactions initiated by a specific sender address.

These conditions are examples of the security notions presented in Section 3.3. We note that the adversary need not be able to decrypt the embedded data to win or, for items (2)–(5), even know of the stegotext's existence. For (1), it is sufficient for the adversary to identify that a transaction contains an embedded message. For (2)–(5), it suffices for the adversary to change modify transactions (stegotexts or associated data) such that the modification is not detected or to replay, reorder, or drop messages even if unaware of the hidden message's existence.

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* $\text{ASGARD.Embed}_{\mathcal{S}}$*, with available q_E embedding queries. Then,*

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

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 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, $\text{Exp}^{\mathsf{asad}_4}_{\Pi,\mathcal{A}}$ 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 $trans.recv\_addr = \text{Enc}(k_J, pk_J)$. 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 $\mathbf{Adv}^{\mathsf{prf}}_{\Pi_{\text{Enc}},\mathcal{B}_4}(\lambda)$. (More discussion on the bases for this assumption can be found in Chapter 5.)
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 $\mathsf{Embed}(\mathsf{ad}, M, h)$, the challenger sets $M_1 \leftarrow M$ and sets $M_0 \leftarrow I$. It then calls $\mathsf{Enc}(M_0, M_1)$, followed by $\mathsf{Sign}(\mathsf{ad}, c_b)$ on the subsequent output c_b, and returns $(\mathsf{ad}, c_b, \sigma)$ to $\mathcal{A}$, where $\mathsf{ad} = (trans.sending\_addr, context, seq)$. 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 $(u < v) \vee (s \neq sent.s_v) \vee (\mathsf{ad} \neq sent.ad_v)$, 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 $\mathbf{Adv}^{\mathsf{SUF\text{-}CMA}}_{q_{\text{Sign}},q_{\text{Vfy}},\Pi_{Sig},\mathcal{B}_1}(\lambda)$. 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 $\mathbf{Adv}^{\mathsf{IND\text{-}CPA}}_{q_{Enc},\Pi_{\text{Enc}},\mathsf{ad},\mathcal{B}_2}(\lambda)$.

$\square$

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. The ideal blockchain model as a ledger-channel ensures that $\mathcal{A}$ cannot force a dropped block unless they are able to violate the consensus process. Thus, while our construction is able to protect against message dropping due to transaction sequence numbers in Ethereum, it may also be possible to achieve Level-4 ASAD through use of nonces on other platforms.*

4.4 Summary

In this chapter, we presented blockchain as a potential embedding vehicle that could meet the constraints set out by our anonymous steganographic model and our ASAD framework. We first discussed broad reasons why permissionless blockchain protocols are particularly useful in the context of steganographic communications. Furthermore, we introduced a specific embedding scheme construction called ASGARD that used an ideal blockchain environment as our communications channel. This scheme repurposed the receiver address for message embedding. We laid out scheme definitions, an encoding algorithm, an extraction algorithm, and a decoding algorithm. Finally, we presented arguments regarding the correctness of our ASGARD embedding scheme as well as a variety of arguments about its IND-CHA security and its Level-i ASAD security.

THIS PAGE INTENTIONALLY LEFT BLANK

CHAPTER 5: A Real-world Implementation of our Blockchain-based Embedding Construction

In this chapter, we take our embedding algorithm out of the ideal blockchain environment and consider its security with respect to the real-world context and implementation, as well as the security requirements presented in the proof of Theorem 4. [10]

5.0.1 Implementation

The embedding algorithm was implemented on the Ethereum platform in a test network. The operating system used was Ubuntu 20.0.4. We used the *geth* client (Go Ethereum) for our test network. Finally, we used the *web3* Python library [107] to interact with our *geth* client [108] and *pycryptodome* Python library for our cryptographic operations [109] (*Cypto.Cipher* and *Crypto.Util.Padding*). A complete list of the algorithms used for our practical implementation can be found below.

- AES-256 in CTR mode as the encryption algorithm
  (implementation selection, *Crypto.Cipher*)
- Keccak-256 algorithm as the hash algorithm
  (Ethereum specification, *web3*)
- ECDSA as the digital signature algorithm
  (Ethereum specification, *web3*)
- Standard Padding Algorithm, PKCS7 (implementation selection, *Crypto.Util.Padding*)

For our experiment, we extracted the first 4800 bits from *Alice in Wonderland*, encrypted it using AES-256 in CTR mode. In our *pycryptodome* and *web3* implementation, this resulted in a ciphertext of length 4800 bits. We then split the ciphertext into 160 bit blocks, and initiated 30 Ethereum transactions with the 160 bit blocks as the receiving address field. As we used 4800 bits, which is divisible by 160, we did not need to pad our input to meet the

[10]Portions of this chapter will be published by IEEE. Reprinted, with permission, from V. Kanth and B. Hale, "Blockchain-based Authenticated Stego-Channels: A Security Framework and Construction", 2022 IEEE International Conference on Blockchain (Blockchain), August, 2022.

field space requirements. (Otherwise, we could have used the standard padding algorithm from *Crypto.Util.Padding* to pad our message prior to encryption.)

5.1 Practical Security Considerations

Based on the transaction internals from Chapter 2, we now take a closer look at the analyses in Chapter 4. Specifically, we examine the Warden's ability to distinguish between the stegotext and the underlying covertext, as well as how our Ethereum-based scheme performs with regards to the ASAD framework.

Comments on Step (3) of the Proof of Theorem 4

As noted in the proof in Chapter 4, we rely an assumption of the mutual pseudorandomness of ciphertexts and hash outputs. More specifically for the construction, it is important that the output of the symmetric encryption algorithm AES-256 and the output of Ethereum's address generation hashing algorithm Keccak-256 are indistinguishable. For the practicality of this, note that ciphertexts generated by a symmetric encryption algorithm are designed to be indistinguishable from random (i.e., if analyzed for IND-CPA security). Similarly, the pseudorandom properties of the Keccak family have been well-explored [110] [111]. [11]

Now, we take a look at the argument for the proof step that selected variants of the Keccak algorithm (Keccak-256) and AES-256 in CTR mode are indistinguishable from random, and thus indistinguishable from each other. To empirically validate the basis of the argument, we also conducted testing as a basis for the pseudorandom output properties of the Keccak-256 algorithm. We generated 400 million bits for analysis via the NIST randomness suite [112] by taking the Keccak-256 hash of the 24-bit representations of the numbers 0 to 1562499 and appending them together. Our bitstream passed all of the NIST tests as can be seen in Appendix B. This empirical testing, in conjunction with the known pseudorandom properties of the Keccak family, provide evidence that the output distribution of the Keccak-256 algorithm is indistinguishable from random.

With regards to our AES-256 in CTR mode implementation for encryption of our plaintext

[11]For Keccak-256 there is one important historical issue to note. The algorithm used in Ethereum is Keccak-256. This version of the Keccak algorithm was not the version that won the SHA3 competition. There is a small difference in padding but the underlying construction was not changed.

messages, AES is a pseudorandom permutation [113], [114]. For CTR mode with AES-256, as used in our implementation, the IV, counter, and random key are fed as inputs to the AES block cipher and the resulting output is XORed with the plaintext message bits. Thus, since AES is pseudorandom, CTR mode encryption within the address field should also be indistinguishable from random.[12] In conclusion, since both the output of AES-256 in CTR mode and Keccak-256 appear random, they are indistinguishable from each other and thus, $\mathcal{A}$ cannot distinguish between them with better than probability $\frac{1}{2}$.

Comments on Step 5 of Theorem 4

The proof of Theorem 4 notably relies on SUF-CMA security of the signature scheme, due to handling of the stegotext. As noted previously, Ethereum uses ECDSA. The security of the specific curve used in Ethereum, secp256k1 is detailed in [115]. ECDSA is not SUF-CMA secure but is known to be EUF-CMA secure [116]. The issue at question is in the malleability of the signature through conversion function handling, namely for an ECDSA signature (s, t), on a message M, the signature $(-s, t)$ is also a valid signature. Ethereum addresses this malleability in its Homestead hard fork [117] by ensuring that transaction signatures whose signature s-value is greater than secp256k1n/2 [13] (i.e., the maximum value of the secp256k1n curve divided by 2) are now considered invalid. With suitable checks in place, our construction can thus meet the proof requirements.

Comments on Step 6 of Theorem 4

AES is used in CTR mode for the tested implementation of ASGARD.Embed for its security properties, specifically IND-CPA security (IND-CPA security based on AES as a pseudorandom permutation and the CTR operation mentioned above). The use of an IV in CTR mode ensures that even if the same message is hidden in the stegotext at a future point, s will appear different. However, it is worth noting the effects of using a different mode with deterministic encryption such as AES in CBC mode) where s is identical for identical embedded messages. In such a case, it will appear to an eavesdropper that Alice is sending a second transaction to Bob (i.e., Bob's address is identical to that in a prior transaction).

[12]We make this argument apart from IND-CPA security of the mode, as the motivation is the indistinguishability of the ciphertext from the hash output, vs. distinguishability among ciphertexts.

[13]secp256k1n = 0xFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFEBAAEDCE6AF48A03BBFD25E8CD0364141 [118]

It is dependent on the environment whether such a second transaction is problematic for security or not, since it is feasible for a party to send a transaction to a particular receiver more than once.

Ethereum vs. the Embedding Algorithm

It is important to consider how the properties of authentication and confidentiality are achieved in the broader Ethereum context. Specifically, Ethereum uses a public-key signature algorithm, ECDSA, which we leverage for authentication in the proof, while AES-256 in CTR mode is a specific addendum for the stego-scheme. Thus, ASAD_4 security is achieved not simply through a newly introduced embedding scheme, but also by leveraging the blockchain infrastructure.

With regards to transaction replays, reordering, or drops as noted in Chapter 5.0.1, Ethereum's transaction structure also includes a sequence number. This sequence number features in the proof of Theorem 4, where the nonce prevents $\mathcal{A}$ from replaying existing transactions. The Ethereum standard currently specifies the nonce to be 64-bits [104]. In normal operation, each new account in Ethereum starts with a sequence value of zero, which is then incremented at each submitted transaction. In practice, since we require that the sender's identity I is used only once, sequence number wrapping is impossible. Even if used repeatedly, it can be enforced that the use of a particular identity I expires once the sequence number space has been exhausted.

5.1.1 Considerations for Adapting the Embedding Scheme to Other Blockchain Protocols

In our indistinguishability experiment, the Warden knows where in the transaction we may have embedded a hidden message (the Warden chooses `ad`). In practice, stego-embedding may take place in other Ethereum transaction fields, dependent on prior arrangement between the sender and receiver, with the stipulation that it is possible to match the underlying distribution of that field. Thus, in a practical sense, the situation is much bleaker for the Warden.

While our embedding construction is Ethereum-based, it is interesting to explore what would happen if our construction was adapted to other blockchain protocols. For example, Bitcoin does not use nonces in its transactions. Instead, it uses a model based on Unspent Transaction Outputs (UTXO) [119]. Therefore, the same embedding scheme would require modification if used on the Bitcoin blockchain, for example, by placing a nonce as a part of the embedded data, which reduces the available bit-length of the text hiding field. In other consensus-based blockchain algorithms that are not instantiated with a sequence number like Ethereum, protection in the form of Level-3 and Level-4 ASAD can be achieved through ensuring that only one embedded transaction appears in a block and waiting until that block has been added to the blockchain before submitting the next transaction. This follows from the ideal blockchain model (as discussed in Chapter 3) where `ledger` as the ledger-channel models the appearance of a transaction on the blockchain before another transaction is computed. In such an implementation, altering the transaction order or dropping a transaction after being added to `ledger` would entail $\mathcal{A}$ being able to violate the consensus protocol. Specifically, $\mathcal{A}$ would need to consistently create blocks, which would imply the ability to violate Byzantine agreement. As noted in Chapter 2, PoW has been shown to reach Byzantine agreement. Thus, practically speaking in a PoW blockchain, consistently creating blocks would require $\mathcal{A}$ to control 51% of the processing power in the blockchain network [65]. While this is not theoretically impossible, it would require a tremendous investment on the part of the adversary.

5.2 Selecting Values for the Associated Data Fields

As an important practical note, Alice *must* take into account typical values in the associated data fields so as to not attract further scrutiny from the Warden. This mirrors AEAD, where the associated data content should not leak information about the encrypted message. For ASGARD, the distribution of the $context$ in `ad` should not be significantly different from that of normal (non-stego) activity. For example, in Ethereum, monetary transactions must be chosen to reduce potential salience. In practice, this would require looking at the distribution of other fields in `ad` and choosing an appropriate value. We now examine some of the associated data fields in our Ethereum-based construction.

5.2.1 Value Field

We provide an example of this type of analysis for the *Value* field in an Ethereum transaction. As our embedded transactions are non-contract transactions, we pulled all of the non-contract transactions that occurred between January of 2021 and February of 2022 and examined their *Value* fields. We used the Google BigQuery ethereum_blockchain database to acquire this data. Google runs a full Ethereum node and records the details of all transaction activity in their database. We performed this analysis with statistical outliers, without statistical outliers, and without statistical outliers with zero elements removed. Table 5.1 shows the relevant summary statistics and Figure 5.1 and Figure 5.2 show the 10-bin histograms of the *Value* fields without outliers for the respective months.

We removed zeros to facilitate the distribution fitting and ensure comparisons between the distributions as some of the distributions are not defined at zero. Furthermore, the proportion of transactions with zero in the *Value* filed is very low. For example, in this February data, 1.3% of the transactions contain zero. More generally, in the context of our steganographic scheme, using a zero-value transaction would be noticeable. Intuitively, if Alice sent Bob an embedded transaction with no money attached to it, it would seem odd to the Warden. This is especially true in Ethereum as transactions require a transaction fee.

With regards to outliers, they skewed both our basic statistics and our ability to fit a distribution to the data. In a practical sense, outliers are of little interest for our steganographic applications as our goal is to hide where there is other activity or noise. In the case of our data distributions, there is far more value in analyzing our data without the outliers.

Table 5.1. Summary Statistics for January 2021- February 2022 Ethereum Non-Contract Transactions *Value* Field

	Outliers (Jan)	No Outliers (Jan)	No Outliers and Zeros (Jan)	Outliers (Feb)	No Outliers (Feb)	No Outliers and Zeros (Feb)
Min	0	0	1.00E-18	0	0	1.00E-18
Max	1500000.000	0.461	0.461	251122.472	0.456	0.456
Mean	7.899	0.083	0.087	3.313	0.089	0.091
Std	562.047	0.100	0.101	228.255	0.101	0.101
Median	0.089	0.045	0.049	0.092	0.049	0.050
IQR	0.442	0.105	0.107	0.252	0.107	0.106

	Outliers (Mar)	No Outliers (Mar)	No Outliers and Zeros (Mar)	Outliers (Apr)	No Outliers (Apr)	No Outliers and Zeros (Apr)
Min	0	0	1.00E-18	0	0	1.00E-18
Max	252800.000	0.431	0.431	450000.000	0.347	0.347
Mean	4.486	0.083	0.086	4.461	0.070	0.073
Std	236.552	0.090	0.090	298.868	0.074	0.074
Median	0.091	0.050	0.052	0.073	0.045	0.048
IQR	0.334	0.089	0.089	0.245	0.085	0.084
	Outliers (May)	No Outliers (May)	No Outliers and Zeros (May)	Outliers (Jun)	No Outliers (Jun)	No Outliers and Zeros (Jun)
Min	0	0	1.00E-18	0	0	1.00E-18
Max	590938.850	0.294	0.294	306714.851	0.241	0.241
Mean	5.770	0.061	0.064	4.248	0.048	0.051
Std	446.568	0.066	0.066	263.014	0.057	0.057
Median	0.059	0.036	0.041	0.046	0.020	0.023
IQR	0.192	0.090	0.088	0.143	0.090	0.095
	Outliers (Jul)	No Outliers (Jul)	No Outliers and Zeros (Jul)	Outliers (Aug)	No Outliers (Aug)	No Outliers and Zeros (Aug)
Min	0	0	1.00E-18	0	0	1.00E-18
Max	600000.000	0.266	0.266	599999.000	0.341	0.341
Mean	3.743	0.054	0.056	3.753	0.068	0.071
Std	331.882	0.062	0.062	266.853	0.077	0.077
Median	0.050	0.025	0.030	0.068	0.036	0.040
IQR	0.152	0.097	0.096	0.194	0.094	0.094
	Outliers (Sep)	No Outliers (Sep)	No Outliers and Zeros (Sep)	Outliers (Oct)	No Outliers (Oct)	No Outliers and Zeros (Oct)
Min	0	0	1.00E-18	0	0	1.00E-18
Max	657334.000	0.410	0.410	420000.000	0.355	0.355

Mean	4.177	0.081	0.083	3.215	0.075	0.077
Std	361.165	0.089	0.089	228.663	0.079	0.079
Median	0.084	0.048	0.050	0.075	0.047	0.049
IQR	0.235	0.097	0.097	0.206	0.092	0.092
	Outliers (Nov)	No Outliers (Nov)	No Outliers a nd Zeros (Nov)	Outliers (Dec)	No Outliers (Dec)	No Outliers and Zeros (Dec)
Min	0	0	1.00E-18	0	0	1.00E-18
Max	345000.000	0.362	0.362	231212.445	0.301	0.301
Mean	3.287	0.078	0.080	3.523	0.068	0.069
Std	302.484	0.078	0.078	249.876	0.073	0.073
Median	0.079	0.051	0.053	0.062	0.040	0.041
IQR	0.202	0.090	0.089	0.192	0.089	0.089
	Outliers (Jan)	No Outliers (Jan)	No Outliers and Zeros (Jan)	Outliers (Feb)	No Outliers (Feb)	No Outliers and Zeros (Feb)
Min	0	0	1.00E-18	0	0	1.00E-18
Max	225698.000	0.476	0.476	600888.000	0.491	0.491
Mean	3.653	0.101	0.103	5.764	0.094	0.098
Std	227.504	0.106	0.106	314.858	0.102	0.103
Median	0.099	0.063	0.066	0.100	0.054	0.057
IQR	0.272	0.114	0.116	0.422	0.101	0.102

For our analysis, we preferred the data without outliers as we are concerned with blending into normal-seeming activity. The outliers greatly affected our summary statistics and our ability to analyze the distribution of the *Value* field. Note that the unit for Ethereum is Ether, which is trading around 3000$ for 1 Ether at the time of writing. Based on this analysis (without outliers), for February of 2022, the sender would want to select a value between 0.0134 ETH (~$40) and 0.120 (~$360), which delineate the 25% quantile and 75% quantile respectively. In contrast, for January of 2022, the sender would want to select a value between 0.021 ETH (~$63) and 0.135 (~$406), which delineate the 25% quantile and 75% quantile respectively. Clearly, the target value can change from month to month. The sender must ensure that they are using the appropriate data in the value field to prevent the Warden from becoming suspicious regarding a potential transaction.

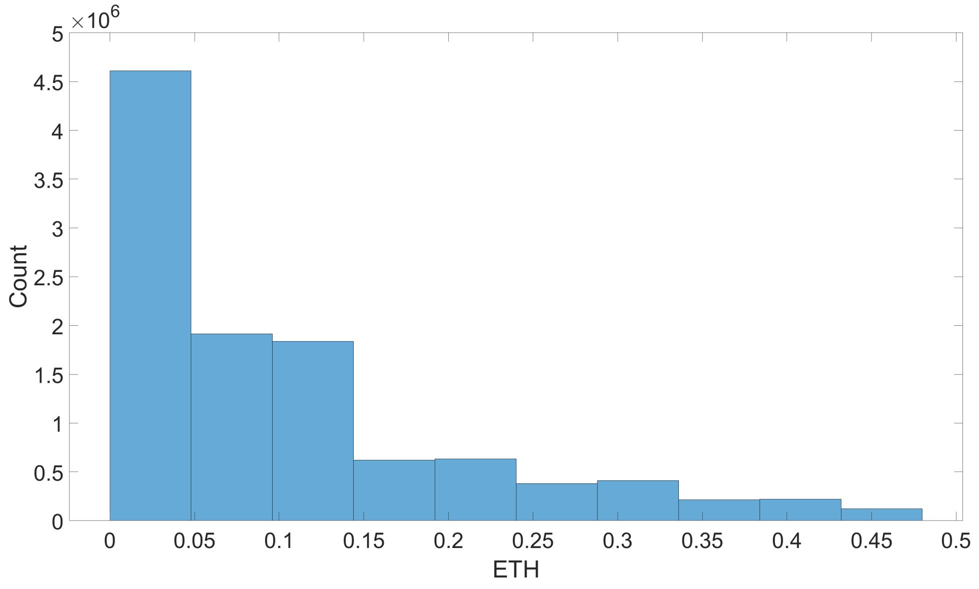

Figure 5.1. Histogram of *Value* Field Counts for January 2022 Ethereum Non-Contract Transactions

In addition to simply observing our basic statistics, we can also fit a distribution to this data. As will also be done in Chapter 6, we used the *fitmethis* [83] function in MATLAB and the *distribution fitter* application in MATLAB to determine the best fit distributions for our data. Chapter 2, provides an overview of MLE, LL, and AIC. The results for all non-contract transactions, with outliers and zeros removed, for February of 2022, can be seen in Table 5.2.

The beta distribution was the best fit for our 2022 data. Figure 5.3 was generated using the *Distribution Fitter* [120] application and depicts the PDF of the standard beta distribution superimposed on a histogram of our February data. We can use the beta distribution to determine what our threshold values for transactions should be. Reintroduced from Chapter 2, the PDF of the beta distribution is defined as [89]:

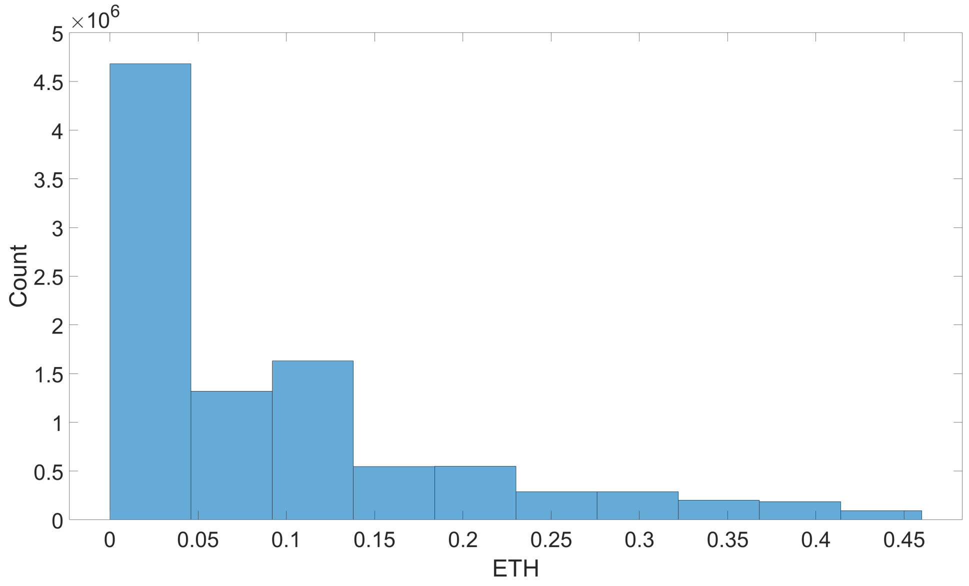

Figure 5.2. Histogram of *Value* Field Counts for February 2022 Ethereum Non-Contract Transactions

$$f(x \mid a, b) = \frac{1}{B(a, b)} x^{a-1}(1-x)^{b-1} \tag{5.2.1}$$

where a is the first shape parameter, b is the second shape parameter, and $B(\cdot)$ is the Beta function. The generalized beta distribution also takes two additional parameters: a lower bound and an upper bound. In the standardized beta distribution, these are 0 and 1 respectively. Now, we can use the CDF of the beta function to determine our threshold values. The CDF of the beta distribution is defined as:

$$F(x \mid a, b) = \frac{1}{B(a, b)} \int_0^x t^{a-1}(1-t)^{b-1}\, dx \tag{5.2.2}$$

where t is the t-distribution. Using the inverse of the beta CDF we can calculate our threshold values (in this case we would like to select values between $p = 0.25$ and $p = 0.75$). Our results from the beta distribution with $a = 0.618$ and $b = 6.22$ are 0.016 and 0.1300 respectively, which line up well with our empirical results. Figure 5.4 shows both the CDF of the beta distribution with the relevant parameters and the CDF of our February data.

Table 5.2. Best Fit Distributions and Corresponding Parameters for *Value* Field, February 2022

Name	Param 1	Param 2	Param 3	LL	AIC
beta	6.18E-01	6.22E+00		1.42E+07	-2.84E+07
nakagami	2.65E-01	1.85E-02		1.42E+07	-2.83E+07
gamma	6.58E-01	1.38E-01		1.41E+07	-2.83E+07
weibull	7.92E-02	7.72E-01		1.40E+07	-2.80E+07
gp	2.24E-01	7.18E-02		1.36E+07	-2.72E+07
exponential	9.08E-02			1.35E+07	-2.69E+07
loglogistic	-3.12E+00	9.61E-01		1.30E+07	-2.61E+07
gev	9.17E-01	3.48E-02	2.65E-02	1.25E+07	-2.50E+07
lognormal	-3.33E+00	2.21E+00		1.07E+07	-2.14E+07
tlocationscale	5.56E-02	5.70E-02	2.21E+00	9.21E+06	-1.84E+07
logistic	7.39E-02	5.25E-02		8.86E+06	-1.77E+07
normal	9.08E-02	1.01E-01		8.40E+06	-1.68E+07
uniform	1.00E-18	4.56E-01		7.56E+06	-1.51E+07
ev	1.48E-01	1.29E-01		5.86E+06	-1.17E+07
rayleigh	9.61E-02			3.47E+06	-6.94E+06
rician	6.39E-05	9.61E-02		3.47E+06	-6.94E+06
birnbaumsaunders	1.00E-06	4.10E+03		-1.27E+07	2.54E+07
inversegaussian	9.08E-02	5.99E-14		-1.07E+08	2.15E+08

We analyzed the parameters for the beta distribution over the period between January 2021 and February 2022. Table 5.3 shows those values.

Clearly, the parameters shift over time (in fact, the best fit distribution for some of the months like August of 2021 is actually the Nakagami distribution). Thus, the prisoners must recalculate their distributions/parameters somewhat frequently. For the best granularity, the prisoners could use the data from the previous day to select the most similar values. From a practical perspective however, dollar amounts in the low hundreds per transaction appear to work well.

Econometrics Observations

Before running the *fitmethis* module in MATLAB, we had expected the Pareto distribution to be our best fit distribution. This was because the Pareto distribution has long been

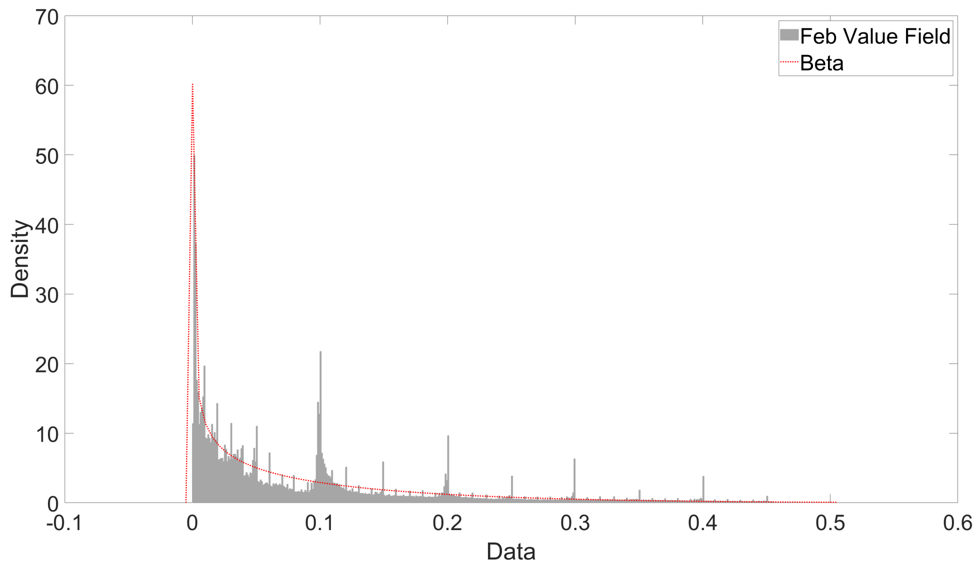

Figure 5.3. Best-fit Distribution for *Value* Field Counts for February, 2022 Ethereum Non-Contract Transactions

used to model income distribution [121] [122] (it was first presented in 1895). However, based on our results there were a number of distributions that performed better than the Pareto distribution for our use case. Looking at the literature, those distributions have also been used for the income distribution use case while performing better than the Pareto distribution. With regards to the Weibull distribution, it was proposed for use with regards to U.S. income data in [123]. In that work, the Weibull distribution was also compared to the Gamma and lognormal distributions. The Gamma distribution was also used to fit U.S. income data in [124]. The beta distribution has also been used for a similar purpose and analyzed in [125]. While we were not able to find literature with regards to the Nakagami distribution, as it is a special case of the Gamma distribution and is commonly used to model right-skewed, positive data sets, its excellent performance in our application makes sense [126]. We also found [127], in which the authors describe using the generalized beta distribution to model income data and compare it to a number of our well-performing distributions. Unfortunately, as MATLAB does not support the generalized beta distribution at this time, and we were able to get practical results for our blockchain embedding scheme using other distributions, we did not test it.

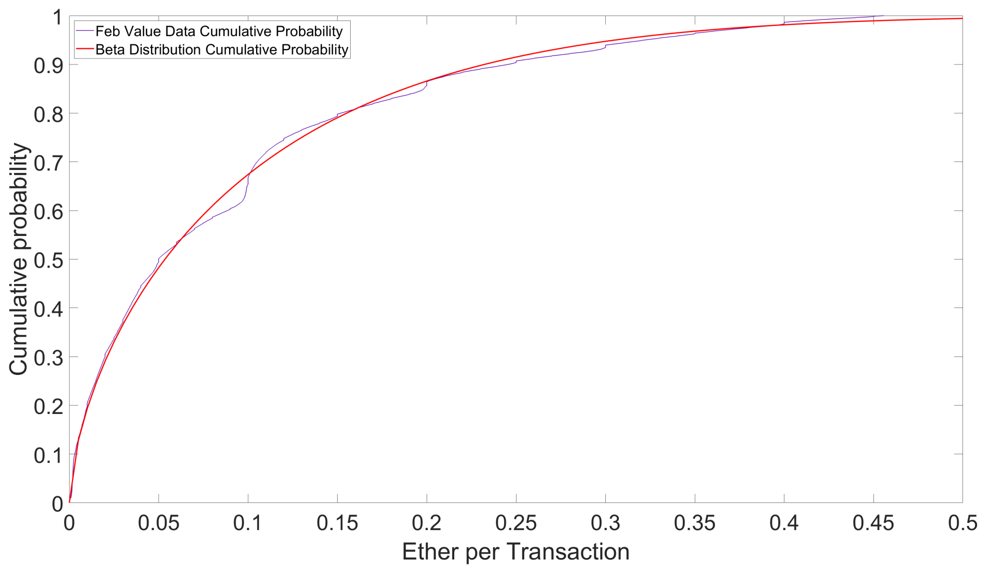

Figure 5.4. Best-fit Distribution for *Value* Field Counts for February, 2022 Ethereum Non-Contract Transactions

5.2.2 *Input Data* Field

We now present an example of poorly selecting a value for a data field in Ethereum. As noted in Chapter 2 embedding data in the *Input Data* field in Ethereum or similar fields in other cryptocurrencies has been explored in the past. Of course, embedding in this field is transparent to Warden and therefore is not particularly covert. That begs the question, how bad is it to embed data in this field? In other words, how much easier does it make the Warden's job in identifying potentially suspicious transactions (i.e., communications from Alice to Bob)?

In Ethereum, the *Input Data* field is primarily used to list smart contract data. It can be used to pass messages between parties. In fact, there are organizations that repurpose that field for their business use case. For example, the company Zoho Sign uses the *Input Data* field to store the hash of a signed document [128]. However, as the *Input Data* field is primarily used for smart contract data, it is rare for transactions to use that field for non-smart contract purposes. To determine how rare these occurrences were, we again used the Google BigQuery database. For every day from December of 2021 to February of 2022, we aggregated three counts. The first was the total number of transactions on that day. The

Table 5.3. Beta Distribution Parameters over January 2021 - February 2022

Month	a	b
Jan	0.0545	0.9418
Feb	0.0618	0.94
Mar	0.0647	1.0726
Apr	0.0603	1.1707
May	0.0469	1.0926
Jun	0.0487	1.3443
Jul	0.0515	1.2917
Aug	0.0532	1.0975
Sep	0.0739	1.2021
Oct	0.0913	1.4954
Nov	0.1029	1.5951
Dec	0.1118	1.9089
Jan	0.1041	1.2814
Feb	0.1151	1.5213

second was the number of transactions that fulfilled the condition $input \neq 0$x (*Input Data* field "not null"). Finally, the third was the number of transactions that fulfilled the "not null" *Input Data* field condition and also a further condition that the recipient address was not a smart contract address (this was accomplished by cross-referencing with the bigquery-public-data-.crypto_ethereum.contracts database). A depiction of this data can be seen in Figure 5.5.

Looking at a single day, in this case January 5th, 2022, there were 1249628 total transactions, 763945 transactions with $input! = 0$x, and only 3185 transactions that had $input! = 0$x and were not sent to a smart contract. Therefore, if Alice was to use the *Input Data* field as their communication mechanism, the Warden would only have to analyze $\frac{3185}{1249628}$ or approximately .25% of the total transactions during that day. This trend continues across the 90 days that we pulled data. Thus, using the *Input Data* to communicate covertly would be foolish. This also has implications for our embedding scheme. We would want to ensure that our *Input Data* was equal to 0x or potentially setup a dummy contract to act as our recipient (though this would alter our scheme from Chapter 5).

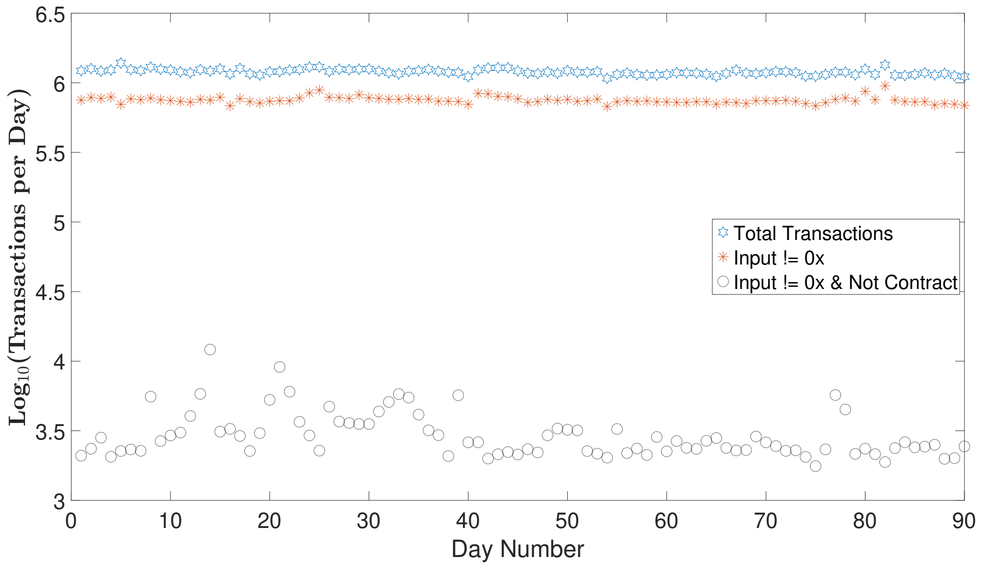

Figure 5.5. Comparison of Number of Transactions, Number of Transactions with "not null" *Input Data* Field, and Number of Transactions with "not null" *Input Data* Field and also not a Smart Contract

5.3 Summary

In this chapter, we presented an Ethereum-based construction of our generic embedding algorithm from Chapter 4. We then analyzed the scheme using our ASAD framework and the specific properties of Ethereum transactions and the Ethereum blockchain structure. We showed that our construction was both IND-CHA secure and Level-4 ASAD secure. Next, we considered our associated data ad and some of the various fields that it contained. Finally, we investigated the properties of the *Value* field and the *Input Data* field and presented appropriate values for use in our Ethereum construction.

THIS PAGE INTENTIONALLY LEFT BLANK

CHAPTER 6: Methods for Detecting Embedded Transactions

As shown in Chapter 4, our Ethereum-based covert communications scheme ASGARD, met our security objectives. One of the key takeaways from the analysis of our scheme was that there are other potential methods for the Warden to detect potentially malicious activity in the Ethereum environment. For example, in Ethereum, transactions that interact with smart contracts can be isolated for further inspection. For most cryptocurrencies, the processing fee associated with transactions can be mapped to detect anomalies. As noted in Chapter 5, the total amount transferred between accounts or the total number of coins (not specific to one cryptocurrency) are other parameters that can be considered. In our specific scheme, if the embedded transaction uses a non-standard amount of gas or transfers no money, this might indicate to an observer that something fishy was going on. We again emphasize that in addition to our embedding field, we also must match the distributions for the other fields in the Ethereum transaction. Even if this is the case, there are other aggregate parameters that the Warden might use to detect suspicious transactions. Intuitively, if a large amount of data must be sent covertly, a large number of transactions must also take place. This observation provides opportunities to the Warden to detect suspicious activity if the embedding parties are not careful. We present the following parameters as examples of this idea. We again used the Google BigQuery ethereum_blockchain database to acquire our data. [14] [15]

6.1 Volumetric Transaction Rate

Regardless of what blockchain-based platform is used, the number of transactions that takes place in a specific time interval is a parameter that can be leveraged. As the hallmark of blockchain is the distributed ledger, both the embedding parties and the Warden have the ability to examine the total number of transactions and the specifics of any individual

[14]Portions of this chapter will be published by IEEE. Reprinted, with permission, from V. Kanth and B. Hale, "Blockchain-based Authenticated Stego-Channels: A Security Framework and Construction," 2022 IEEE International Conference on Blockchain (Blockchain), August, 2022.

[15]Portions of this chapter were previously published by Springer [3]. Reproduced with permission from Springer Nature. Material from: Kanth, V., McEachen, J., Tummala, M. (2022). Parameter Identification for Malicious Transaction Detection in Blockchain Protocols. In: Prieto, J., Partida, A., Leitão, P., Pinto, A. (eds) Blockchain and Applications. BLOCKCHAIN 2021. Lecture Notes in Networks and Systems, vol 320.

transaction. The goal from the Warden's perspective is to reduce the total number of transactions that must be investigated to a manageable number. One way to do this is to investigate periods of time in which a statistically large number of transactions takes place.

In the context of our embedding scheme, this might correspond to Alice sending Bob a long message during a short duration of time. In our ASGARD construction, we embed 160 bits per transaction. In order to send a message that is 10 kB, we would need to send 512 embedded transactions. If we were to do so in a short period of time, we could impact the distribution of the transaction traffic. Thus, the Warden must thoroughly understand and analyze the underlying distribution of the transaction traffic for our platform of choice to detect our embedding activity. As we studied Ethereum transaction traffic, we noticed that its distribution appeared to shift over time. An example of this can be seen in Figure 6.1, which shows transaction counts in 10-minute intervals for two different months. Not only did the shape vary, the location and scale of the peak also varied. Thus, we first explored the stationarity of Ethereum traffic.

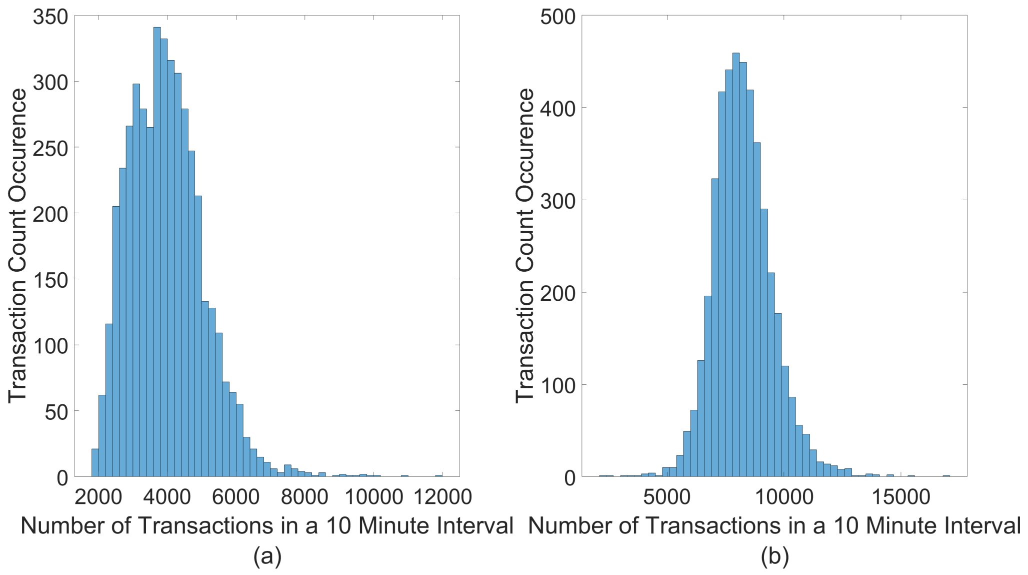

Figure 6.1. Transaction Counts Comparison between Jan 2020 (a) and Jan 2022 (b)

Not surprisingly, due to the relative youth of the Ethereum platform the transaction traffic distribution appears non-stationary or, in other words, it changes over time. In order to provide evidence of this assertion, we used a variety of statistical tests to examine Ethereum daily transaction data. We pulled this data from etherscan.io [129] and performed our analysis in MATLAB. A figure graphing the transaction data can be seen in Figure 6.2.

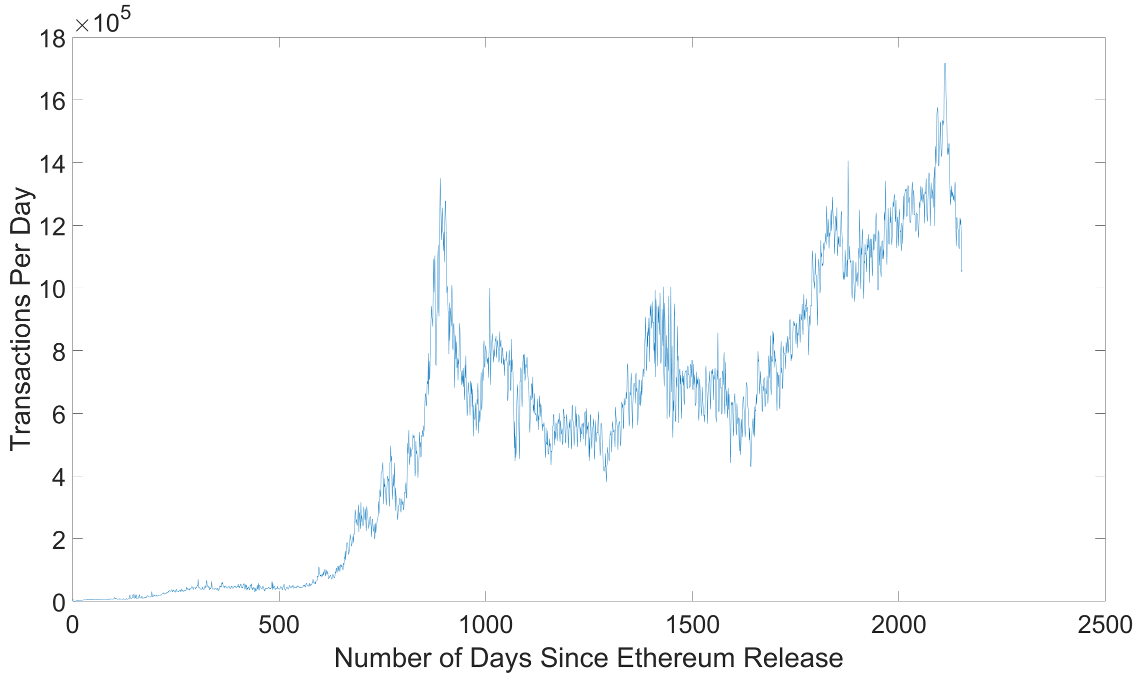

Figure 6.2. Ethereum Daily Transactions Chart, data from [129]

To test the non-stationary nature of the Ethereum daily transaction data, we used the Augmented Dickey-Fuller (ADF) test and the Kwiatkowski–Phillips–Schmidt–Shin (KPSS) test. The ADF test tests for the presence of a unit root. A unit root is a process whose first difference is stationary. It has the form [130]:

$$y_t = y_{t-1} + x$$

where x is a stationary process. The null hypothesis for the ADF test is that there is a unit root. The alternative hypothesis is that there is no unit root, or that the series is stationary. From [131], the ADF test uses the model:

$$y_t = c + \delta t + \phi y_{t-1} + \ldots + B\Delta y_{t-p} + \epsilon_t$$

where Δ is the differencing operator ($\Delta y = y_t - y_{t-1}$), p is the number of lagged difference terms, and ϵ_t is a mean zero innovation process. In this case, the null hypothesis and alternative hypothesis are

$$H_0 : \phi = 1$$
$$H_A : \phi < 1$$

For the number of lags, we used the recommendation by Schwert in [132] to set the max number of lags as $\lfloor 12(T/100)^{1/4} \rfloor$. For this particular dataset, as we adjusted the number of lags, the end result did not change. We ran the ADF test in MATLAB and we failed to reject the null hypothesis and concluded the distribution was non-stationary. To confirm this result, we also ran our data through the KPSS test, which tests whether a time series is trend stationary versus that it is a non-stationary unit process. It uses the model [133]:

$$y_t = c_t + \delta t + u_{1t}$$
$$c_t = c_{t-1} + u_{2t}$$

where δ is the trend coefficient, u_{1t} is a stationary process, and u_{2t} is an independent and identically distributed process with mean 0 and variance σ^2. The null hypothesis and alternative hypothesis are:

$$H_0 : \sigma^2 = 0$$
$$H_A : \sigma^2 > 0$$

The null hypothesis implies that c_t, the random walk term is constant, whereas the alternative hypothesis indicates the existence of the unit root in the random walk. The test statistic is:

$$\frac{\sum_{t=1}^{T} S_t^2}{s^2 T^2}$$

where T is the sample size, s^2 is the Newey-West estimate of the long-run variance, and $S_t = e_1 + e_2 + \ldots E_t$. The test also requires a suitable number of lags. Based on [134], a number of lags on the order of $\sqrt{T}$, where T is the sample size, is appropriate. We performed the KPSS test in MATLAB and got a p-value of .01 allowing us to reject our null-hypothesis. This result, in conjunction with our result from the ADF test, provides good evidence that this data is indeed non-stationary. This fact proved to be very significant in our analysis of specific time intervals.

Based on the time period of interest, for example a single day, that day can be further subdivided into smaller time intervals (e.g., 10-minute intervals) for more granular analysis. Once time intervals of interest are determined, the specific transactions within that interval can be examined using a variety of techniques or filtered even further using other parameters. Ideally, we would want an idea of what typical activity looked like for a specific period of time in order to identify anomalous time periods. Based on our knowledge that Ethereum is a maturing platform, and that its traffic distribution in non-stationary, our goal in this parameter analysis is two-pronged. First, we want to determine what distribution type best captures transaction activity in more granular time intervals as to best inform analysis of the present time. Secondly, we want to show that the distribution of these granular time intervals has also changed over time. In order to accomplish this, we counted the number of transactions that occurred in 10-minute intervals over several months. We then took those transaction counts and graphed them in a histogram. On the subject of time interval selection, we decided to select time periods that had occurred recently. This was because these time intervals were better predictors of modern activity than earlier data. As Figure 6.2 shows, very few transactions occurred during the first year post-Ethereum release and we are interested in modern illicit activity. To that end, we began our analysis with data from 2020. Figure 6.3 shows a comparison between January 2020-March 2020 (a) and January 2021-March 2021 (b).

Obviously, the number of transactions that occurred was much greater in 2021 than in 2020. The shape of the distributions also looks different. In order to assess the differences between these distributions, we used the *fitmethis* [83] function in MATLAB. We tested different types of transformations on the number of transactions in a 10-minute interval axis before fitting a model to our distribution. These transformations (see [135]) were a normalization (divide each x-axis value by the max x value), the log transform (each x value is replaced by $\log(x)$), and the reciprocal transform (each x value is replaced by $\frac{1}{x}$).

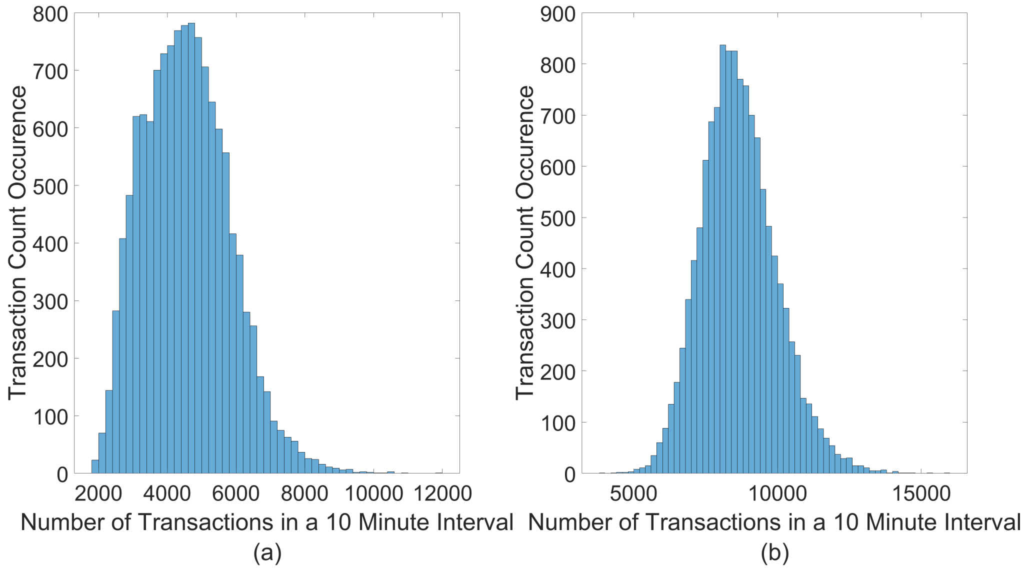

Figure 6.3. Transaction Counts Comparison between Jan-Mar 2020 (a) and Jan-Mar 2021 (b)

Figure 6.4 demonstrates an example of this distribution fitting process and has been truncated to better illustrate the differences between the respective distributions. Table 6.1 is a tabulation of the relevant parameters, LL, and AIC for each of the distributions. The time period chosen was Jan 2022 and the transform used was the reciprocal transform. As noted in Chapter 5, if the AIC for one model is more than 2 less than another, that model is superior [87]. For example, from Table 6.1, the AIC for the loglogistic distribution is −84671, which is 99 less than the AIC for the tlocationscale distribution −84572. Thus, the loglogistic distribution is a better fit for our Ethereum transaction data for Jan 2022.

A summary of the best fit distributions over one-month periods from September of 2020 to June of 2022 can be found in Table 6.2. We also used a sliding window over three-month intervals to better understand the effects of individual months on the aggregate distribution. Those results can be seen in Table 6.3. The time period of study was from September of 2020 to June of 2022.

These tables reveal a couple of important trends. The distribution that best fit the transaction data has changed over our studied time period. Another important observation was that the

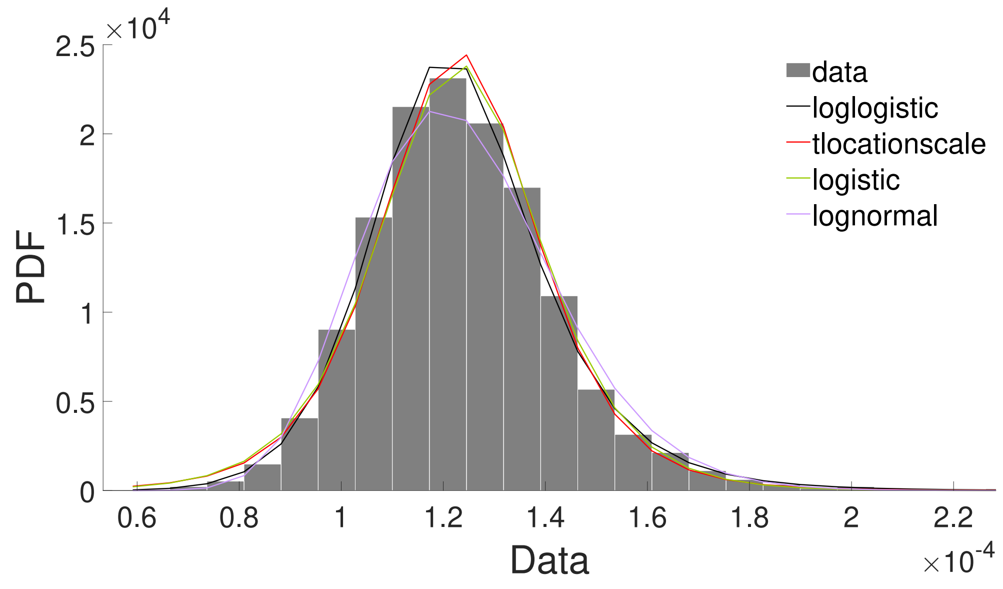

Figure 6.4. Comparison of Different Distributions for Jan 2022

three-month sliding window results were better from a LL and AIC perspective than the one-month results. Furthermore, the reciprocal transform provided the best fit across all of our data. From an analysis perspective, having the most accurate distribution at the desired time of analysis is significant. For example, finding a period of time where Alice and Bob embedded hidden messages would require the Warden to understand the distribution of transactions at a granular level. One of the advantages the Warden has in a blockchain-based environment is that the ledger is eternal. They can go back in the transaction history to potentially find hidden data. Of course, as noted in Chapter 3 the content of the hidden communications between Alice and Bob might be expired, i.e., they may have already escaped the prison. Furthermore, the fact that the traffic distribution appears to be non-stationary means that until the Ethereum platform reaches a steady-state condition, the Warden must continuously update and study these models. To illustrate this point, Figure 6.5 shows the best fit distribution comparison between reciprocal-transformed transaction data for (a) January 2020-March 2020 and (b) January 2021-March 2021.

To demonstrate how the Warden might use knowledge of the traffic distribution to reduce the overall number of transactions to inspect, we picked a target time period of Jan-Mar 2021. For this time period, the best fit distribution was log-normal as can be seen in Figure 6.6. It

Table 6.1. Distribution Parameters and Fit for Jan 2022 Data:

Name	Par1	Par2	Par3	LL	AIC
loglogistic	-9.01E+00	8.41E-02		4.23E+04	-8.47E+04
tlocationscale	1.23E-04	1.56E-05	5.79E+00	4.23E+04	-8.46E+04
logistic	1.23E-04	1.05E-05		4.23E+04	-8.45E+04
lognormal	-9.01E+00	1.54E-01		4.22E+04	-8.45E+04
inversegaussian	1.24E-04	5.17E-03		4.22E+04	-8.44E+04
birnbaumsaunders	1.23E-04	1.55E-01		4.22E+04	-8.44E+04
gev	-3.73E-02	1.72E-05	1.15E-04	4.22E+04	-8.44E+04
gamma	4.15E+01	2.99E-06		4.22E+04	-8.43E+04
beta	4.15E+01	3.35E+05		4.22E+04	-8.43E+04
nakagami	1.01E+01	1.58E-08		4.21E+04	-8.41E+04
rician	1.22E-04	2.04E-05		4.19E+04	-8.38E+04
normal	1.24E-04	2.02E-05		4.19E+04	-8.38E+04
weibull	1.32E-04	4.58E+00		4.10E+04	-8.19E+04
ev	1.36E-04	4.82E-05		3.88E+04	-7.77E+04
rayleigh	8.89E-05			3.86E+04	-7.72E+04
gp	-3.50E-01	1.48E-04		3.65E+04	-7.29E+04
exponential	1.24E-04			3.57E+04	-7.14E+04
uniform	5.91E-05	4.23E-04		3.54E+04	-7.07E+04
Kernel	3.43E-06			4.24E+04	1.04E+09

is important to note that though the loglogistic distribution was the best fit for the majority of our time intervals, as we were concerned with our specific time period of Jan-Mar 2021 we used the log-normal distribution. More broadly, if the Warden was looking to characterize activity today, they would be better served using the loglogistic distribution as it was the best fit distribution for the last several sliding windows.

Returning to our time period of Jan-Mar 2021, we can use the parameters of the distribution to highlight suspicious time intervals. For example, as referenced from Chapter 2, the PDF of the log-normal distribution is [88]:

$$f(x \mid \mu, \sigma) = \frac{1}{x\sigma\sqrt{2\pi}} \exp\left\{\frac{-(\log x - \mu)^2}{2\sigma^2}\right\}, \text{for } x > 0 \qquad (6.1.1)$$

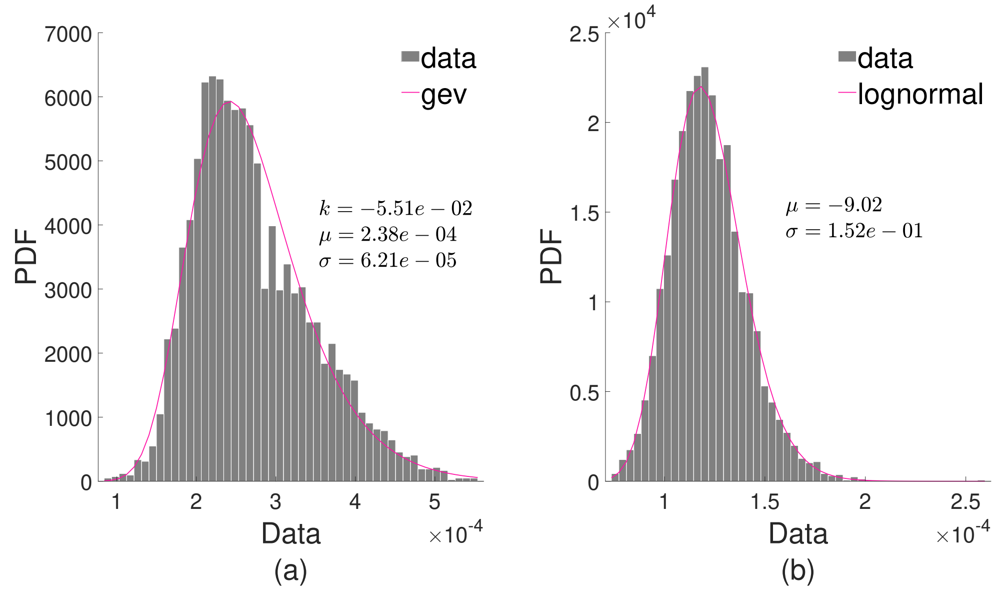

Figure 6.5. Distribution Comparison between Jan-Mar 2020 (a) and Jan-Mar 2021 (b)

The parameters of interest for this distribution are the mean μ, and the standard deviation σ. The best fit distribution during the time period Jan-Apr 2021 has $\mu = -9.0560$ and $\sigma = 0.1507$.

Figure 6.6 shows the log-normal distribution fitted to the transaction data from Jan-Apr 2021 with σ and $2 \cdot \sigma$ marked. These regions can be used to select time periods where the number of transactions falls well outside what would be expected based on the distribution and reduces the total number of suspicious transactions that must be analyzed further. In this dataset, $2 \cdot \sigma$ corresponds to 10-minute intervals in which more than 11590 transactions occurred. Any time interval with more transactions could be flagged as suspicious, greatly reducing the overall number of transactions that need to be deeply inspected. In our data, only 290 intervals out of 12816 total intervals, or 2.26% of intervals, had more than 11590 transactions occur.

The challenge facing the Warden is complicated by real-world complexities, however. In ASGARD, a 10-minute interval is likely too fast to capture the range of embedding activity. As noted before, sending 1 kB of data would require 512 transactions. Even if all of those

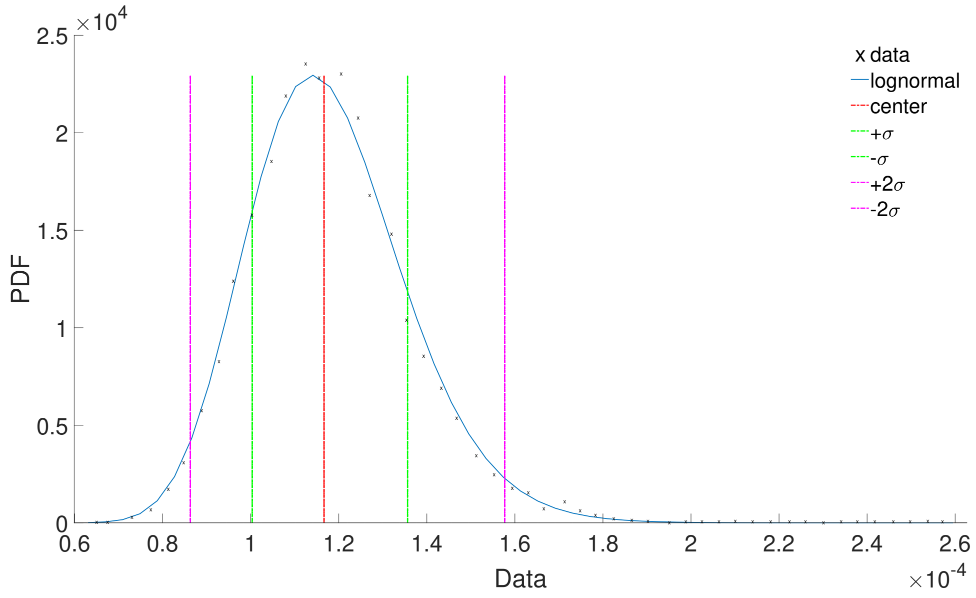

Figure 6.6. Log-normal Distribution of Jan-Mar 2021

transactions were executed at the same time, depending on the gas price and network congestion, it could take longer than 10 minutes to process of all of them. Thus, this means that the Warden must look at a wider time interval or use another parameter in conjunction with volumetric transaction rate.

6.2 Unique Addresses Activity

Another useful method the Warden could use to detect potential embedding activity is to look at the distribution of entity activity. In the case of most blockchain protocols, an entity is represented by either a single address or a set of addresses. Knowing that transactions are the vehicle for data embedding, looking at high traffic addresses or sets of addresses can potentially ferret out Alice and Bob. Pham and Lee [136] performed such an analysis on the Bitcoin dataset (from genesis up to April 7, 2013) and used k-means clustering as a method to detect outliers. While they focus on long term data, we specifically target more granular periods of time for our analysis. This facilitates capturing entities that only operate for short periods of time and are summarily discarded, as occurs in our ASGARD construction.

In order to demonstrate how the Warden might use this approach, addresses that initiated transactions were extracted from Ethereum transaction data between March 5, 2021 to March 15, 2021 using the Google BigQuery ethereum_blockchain database. Python3 was used for the visualizations and distribution fitting in this section. A histogram showing the number of transactions associated with the $\log_{10}$ of the number of sending addresses for a 10-day period between March 5-15, 2021 is shown in Figure 6.7.

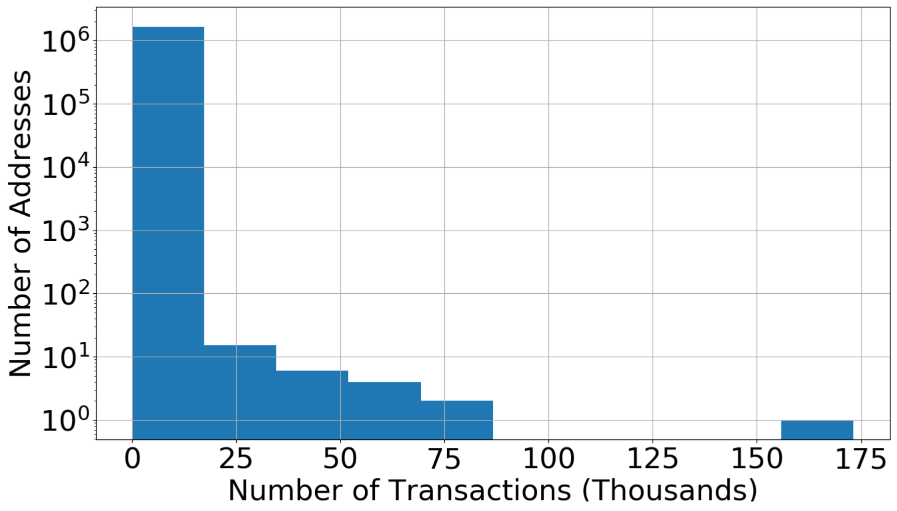

Figure 6.7. Histogram of Number of Transactions versus Number of Addresses from March 5-15, 2021

As might be expected, the vast majority of addresses only initiate a few transactions. This means we can use this data to set a threshold of addresses whose transaction history might be suspicious. Furthermore, we can map a distribution to this data to further inform threshold selection. With this type of network data, logarithmic binning is often used to reduce the number of bins required for a histogram. Once that binning is complete, we used the center of the bin as a discrete data point for curve fitting [137]. Blockchain address data can often be fit to the power law function [136] [138] [139], which is defined as:

$$f(x \mid A, k) = Ax^k \tag{6.2.1}$$

For this function, A is amplitude and k is the exponent. Figure 6.8 shows a power law model being fitted to our 10 day period. The value of A was 7.320 ± 0.195 and the value of k was -0.131 ± 0.006. The R^2 value was 98.8%, indicating that the model was an excellent fit for the data. With these parameters, a threshold can be set whereby accounts that initiate over a certain number of transactions during a particular time interval can be flagged. Then, the transactions initiated by that account can be analyzed further for malicious or illegal activity.

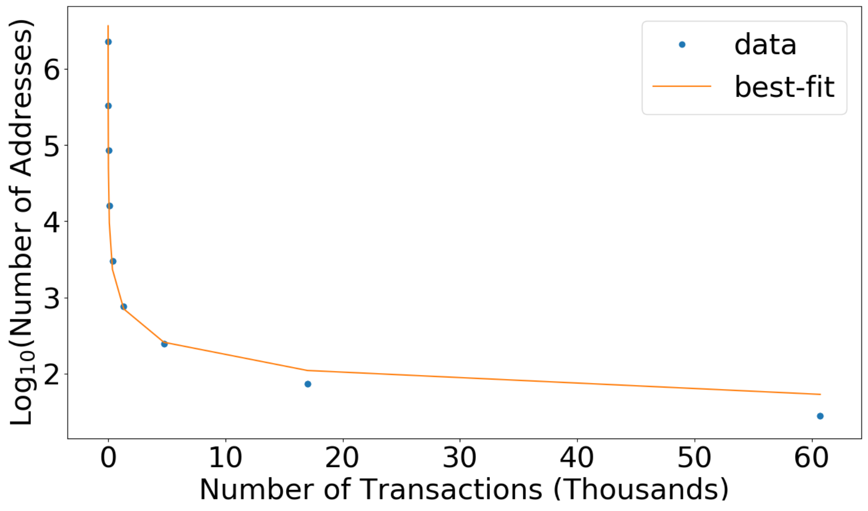

Figure 6.8. Power law Fit of Histogram of Number of Transactions versus Number of Sender Addresses from March 5-15, 2021

Not unexpectedly, looking at another same 10-day period with respect to receiving address instead of sending addresses shows a similar distribution with different parameters. Figure 6.9 shows the power law model being fitted to the receiver address distribution for the time period between March 5-15, 2021.

The value of A was 7.198 ± 0.369 and the value of k was -0.139 ± 0.011. The R^2 value was 96.8% indicating that the model was an excellent fit for this data as well. These distributions can help to reduce the total number of transactions that might be considered suspicious for analysis. Once these transactions have been identified, they can be used as input to a variety of follow-on detection algorithms. For example, perhaps the Warden only looks at addresses that performed more than 20 transactions in a 10-day interval as suspicious. We stress that this trend exists for any individual day and thus can be used in conjunction with our volumetric analysis to further reduce the number of transactions of interest. From the perspective of our ASGARD construction, this type of analysis is the reason why the size of the identity sets is significant. If addresses can be correlated, the sum of transactions from suspicious addresses can be used in a similar manner to potentially detect our embedding

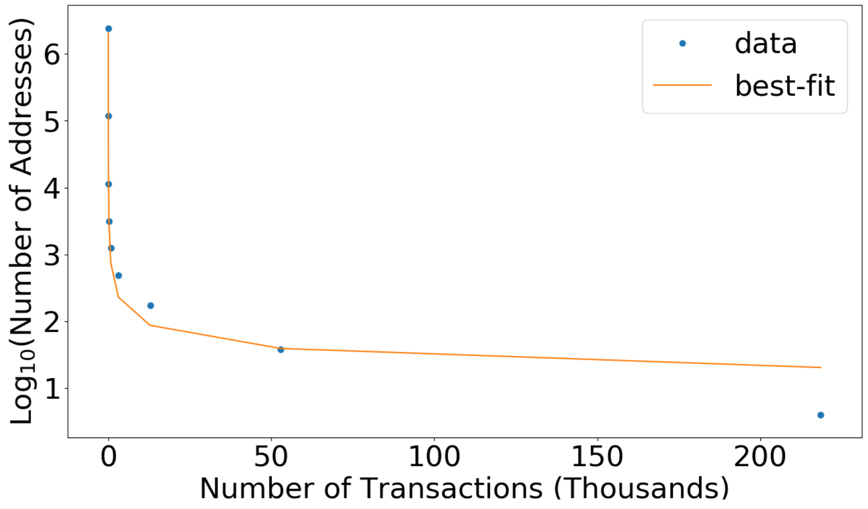

Figure 6.9. Power law Fit of Histogram of Number of Transactions versus Number of Receiver Addresses from March 5-15, 2021

strategy. This was the basis for our recommendation that Alice and Bob only use an address a single time before swapping. Of course, this creates practical hurdles and leads to an interesting avenue of future work, characterizing the number of times a unique address can be used before it becomes suspicious.

6.3 Summary

In this chapter, we presented some parameters that the Warden might be able to use to detect our embedding construction. We presented evidence of the non-stationary nature of the Ethereum distribution, explored volumetric transaction rate and unique address activity as parameters of interest, and showed how those parameters might be used to detect suspicious transactions or account addresses.

Table 6.2. Best-fit Distribution for Transaction Counts over One-Month Intervals From September of 2020 to June of 2022

Sep 20	Norm	Log	Reciprocal	Oct 20	Norm	Log	Reciprocal
Distribution	lognormal	nakagami	lognormal	Distribution	gev	normal	inversegaussian
LL	4.13E+03	4.76E+03	3.97E+04	LL	3.98E+03	5.05E+03	4.11E+04
AIC	-8.26E+03	-9.52E+03	-7.94E+04	AIC	-7.96E+03	-1.01E+04	-8.21E+04
Nov 20	Norm	Log	Reciprocal	Dec 20	Norm	Log	Reciprocal
Distribution	gamma	normal	gev	Distribution	lognormal	tlocationscale	lognormal
LL	4.27E+03	5.18E+03	4.03E+04	LL	5.09E+03	5.40E+03	4.18E+04
AIC	-8.54E+03	-1.04E+04	-8.05E+04	AIC	-1.02E+04	-1.08E+04	-8.35E+04
Jan 21	Norm	Log	Reciprocal	Feb 21	Norm	Log	Reciprocal
Distribution	lognormal	tlocationscale	lognormal	Distribution	gamma	tlocationscale	lognormal
LL	4.27E+03	5.80E+03	4.24E+04	LL	4.06E+03	5.12E+03	3.72E+04
AIC	-8.53E+03	-1.16E+04	-8.47E+04	AIC	-8.11E+03	-1.02E+04	-7.43E+04
Mar 21	Norm	Log	Reciprocal	Apr 21	Norm	Log	Reciprocal
Distribution	lognormal	tlocationscale	lognormal	Distribution	lognormal	tlocationscale	lognormal
LL	4.98E+03	6.02E+03	4.28E+04	LL	5.40E+03	5.35E+03	4.13E+04
AIC	-9.95E+03	-1.20E+04	-8.56E+04	AIC	-1.08E+04	-1.07E+04	-8.27E+04
May 21	Norm	Log	Reciprocal	Jun 21	Norm	Log	Reciprocal
Distribution	gamma	tlocationscale	gev	Distribution	loglogistic	tlocationscale	loglogistic
LL	4.56E+03	5.02E+03	4.24E+04	LL	4.41E+03	4.94E+03	4.03E+04
AIC	-9.11E+03	-1.00E+04	-8.48E+04	AIC	-8.82E+03	-9.87E+03	-8.06E+04
Jul 21	Norm	Log	Reciprocal	Aug 21	Norm	Log	Reciprocal
Distribution	loglogistic	tlocationscale	loglogistic	Distribution	tlocationscale	tlocationscale	loglogistic
LL	4.54E+03	4.96E+03	4.15E+04	LL	4.76E+03	5.38E+03	4.19E+04
AIC	-9.08E+03	-9.92E+03	-8.30E+04	AIC	-9.52E+03	-1.08E+04	-8.37E+04
Sep 21	Norm	Log	Reciprocal	Oct 21	Norm	Log	Reciprocal
Distribution	tlocationscale	tlocationscale	loglogistic	Distribution	loglogistic	tlocationscale	loglogistic
LL	7.45E+03	5.17E+03	4.04E+04	LL	5.08E+03	5.49E+03	4.22E+04
AIC	-1.49E+04	-1.03E+04	-8.08E+04	AIC	-1.02E+04	-1.10E+04	-8.44E+04
Nov 21	Norm	Log	Reciprocal	Dec 21	Norm	Log	Reciprocal
Distribution	loglogistic	tlocationscale	loglogistic	Distribution	loglogistic	tlocationscale	loglogistic
LL	6.79E+03	5.85E+03	4.15E+04	LL	6.45E+03	5.73E+03	4.24E+04
AIC	-1.36E+04	-1.17E+04	-8.31E+04	AIC	-1.29E+04	-1.15E+04	-8.48E+04
Jan 22	Norm	Log	Reciprocal	Feb 22	Norm	Log	Reciprocal
Distribution	loglogistic	tlocationscale	loglogistic	Distribution	loglogistic	tlocationscale	tlocationscale
LL	5.38E+03	5.86E+03	4.23E+04	LL	5.46E+03	5.51E+03	3.84E+04
AIC	-1.08E+04	-1.17E+04	-8.47E+04	AIC	-1.09E+04	-1.10E+04	-7.68E+04
Mar 22	Norm	Log	Reciprocal	Apr 22	Norm	Log	Reciprocal
Distribution	loglogistic	tlocationscale	tlocationscale	Distribution	loglogistic	tlocationscale	tlocationscale
LL	6.81E+03	6.15E+03	4.25E+04	LL	6.51E+03	6.04E+03	4.11E+04
AIC	-1.36E+04	-1.23E+04	-8.51E+04	AIC	-1.30E+04	-1.21E+04	-8.22E+04
May 22	Norm	Log	Reciprocal				
Distribution	loglogistic	tlocationscale	loglogistic				
LL	5.73E+03	5.83E+03	4.21E+04				
AIC	-1.15E+04	-1.17E+04	-8.42E+04				

Table 6.3. Best-fit Distribution for Transaction Counts over Three-Month Sliding Windows From August of 2020 to June of 2022

Aug-Oct 20	Norm	Log	Reciprocal	Sep-Nov 20	Norm	Log	Reciprocal
Distribution	gamma	normal	gev	Distribution	birnbaumsaunders	normal	inversegaussian
LL	1.25E+04	1.47E+04	1.22E+05	LL	1.29E+04	1.49E+04	1.21E+05
AIC	-2.50E+04	-2.94E+04	-2.44E+05	AIC	-2.58E+04	-2.98E+04	-2.46E+05
Oct-Dec 20	Norm	Log	Reciprocal	Nov 20-Jan 21	Norm	Log	Reciprocal
Distribution	lognormal	tlocationscale	lognormal	Distribution	lognormal	tlocationscale	lognormal
LL	1.50E+04	1.54E+04	1.23E+05	LL	1.52E+04	1.61E+04	1.24E+05
AIC	-3.00E+04	-3.08E+04	-2.46E+05	AIC	-3.03E+04	-3.22E+04	-2.48E+05
Dec 20-Feb 21	Norm	Log	Reciprocal	Jan-Mar 21	Norm	Log	Reciprocal
Distribution	lognormal	tlocationscale	lognormal	Distribution	lognormal	tlocationscale	lognormal
LL	1.45E+04	1.59E+04	1.21E+05	LL	1.40E+04	1.68E+04	1.22E+05
AIC	-2.89E+04	-3.18E+04	-2.42E+05	AIC	-2.79E+04	-3.35E+04	-2.44E+05
Feb-Apr 21	Norm	Log	Reciprocal	Mar-May 21	Norm	Log	Reciprocal
Distribution	lognormal	tlocationscale	lognormal	Distribution	lognormal	lognormal	gamma
LL	1.69E+04	1.60E+04	1.21E+05	LL	1.59E+04	1.56E+04	1.26E+05
AIC	-3.37E+04	-3.20E+04	-2.42E+05	AIC	-3.19E+04	-3.11E+04	-2.51E+05
Apr-Jun 21	Norm	Log	Reciprocal	May-Jul 21	Norm	Log	Reciprocal
Distribution	gamma	tlocationscale	lognormal	Distribution	loglogistic	tlocationscale	loglogistic
LL	1.46E+04	1.40E+04	1.23E+05	LL	1.37E+04	1.35E+04	1.23E+05
AIC	-2.91E+04	-2.80E+04	-2.45E+05	AIC	-2.73E+04	-2.69E+04	-2.45E+05
Jun-Aug 21	Norm	Log	Reciprocal	Jul-Sep 21	Norm	Log	Reciprocal
Distribution	loglogistic	tlocationscale	loglogistic	Distribution	tlocationscale	tlocationscale	loglogistic
LL	1.40E+04	1.53E+04	1.24E+05	LL	2.23E+04	1.55E+04	1.24E+05
AIC	-2.80E+04	-3.05E+04	-2.47E+05	AIC	-4.46E+04	-3.09E+04	-2.47E+05
Aug-Oct 21	Norm	Log	Reciprocal	Sep-Nov 21	Norm	Log	Reciprocal
Distribution	loglogistic	tlocationscale	loglogistic	Distribution	tlocationscale	tlocationscale	loglogistic
LL	4.15E+04	1.59E+04	1.24E+05	LL	2.23E+04	1.61E+04	1.24E+05
AIC	-8.31E+04	-3.17E+04	-2.49E+05	AIC	-4.46E+04	-3.21E+04	-2.47E+05
Oct-Dec 21	Norm	Log	Reciprocal	Nov 21-Jan 22	Norm	Log	Reciprocal
Distribution	loglogistic	tlocationscale	loglogistic	Distribution	loglogistic	tlocationscale	loglogistic
LL	2.02E+04	1.69E+04	1.26E+05	LL	2.05E+04	1.70E+04	1.26E+05
AIC	-4.04E+04	-3.38E+04	-2.52E+05	AIC	-4.11E+04	-3.40E+04	-2.52E+05
Dec 21-Feb 22	Norm	Log	Reciprocal	Jan-Mar 22	Norm	Log	Reciprocal
Distribution	loglogistic	tlocationscale	loglogistic	Distribution	loglogistic	tlocationscale	loglogistic
LL	1.94E+04	1.69E+04	1.23E+05	LL	1.92E+04	1.75E+04	1.23E+05
AIC	-3.88E+04	-3.39E+04	-2.46E+05	AIC	-3.85E+04	-3.49E+04	-2.46E+05
Feb-Apr 22	Norm	Log	Reciprocal	Mar-May 22	Norm	Log	Reciprocal
Distribution	loglogistic	tlocationscale	loglogistic	Distribution	loglogistic	tlocationscale	loglogistic
LL	1.96E+04	1.76E+04	1.22E+05	LL	2.02E+04	1.79E+04	1.26E+05
AIC	-3.92E+04	-3.52E+04	-2.44E+05	AIC	-4.03E+04	-3.58E+04	-2.51E+05

THIS PAGE INTENTIONALLY LEFT BLANK

CHAPTER 7: Conclusion

The goal of our work was the exploration of how privacy-based applications and anonymity-based applications like blockchain-based protocols have changed the information hiding space. Broadly speaking, our work fell into three categories. The first category covered the role of anonymity in steganographic schemes, developed a model for anonymized secure communications, and proposed with a framework to assess and evaluate those schemes. In the furtherance of these efforts, we presented a set of definitions and experiments designed to capture the level of security that a scheme exhibited.

The second category revolved around the development of a specific embedding approach (in this case using the receiving address field in a blockchain-based protocol) and a practical implementation of that approach in a current blockchain protocol (Ethereum). Our Ethereum test network served as a mechanism to showcase our practical implementation in a real world scenario. We then performed an analysis of those approaches and implementations using our steganographic security analysis framework, ASAD.

Finally, our third category was an analysis of blockchain protocol data, particularly for the Ethereum platform. We analyzed several different components of Ethereum transactions and developed metrics for malicious transaction detection that properly used, could potentially detect some of our embedding activity. These three lines of effort worked in concert to create a cohesive narrative regarding steganography in an age of anonymity with blockchain as a major example of emerging anonymity-based technologies.

7.1 Significant Contributions

The research encapsulated within this dissertation has advanced the body of research in steganography. Our contribution in this area is twofold as we both consider the role anonymity plays within a steganographic scheme and establish an anonymized secure communications model, as well as provide a framework by which steganographic schemes can be evaluated. Our second significant contribution was the development of blockchain-based embedding algorithms and a practical implementation of those algorithms in an

Ethereum testbed. Furthermore, we provided a security analysis of our embedding algorithms and practical implementation. Our final major contribution was our data modeling of the Ethereum blockchain and the development of metrics for malicious transaction detection and an analysis of their potential impacts on our embedding approaches.

7.1.1 Extending the Body of Steganographic Literature

We identified that Simmons' seminal conception of steganographic channels, the Prisoners' Problem [19], was not sufficient to describe situations where the Warden could not positively determine the identities of their prisoners. To address this shortcoming, we developed the anonymized Prisoners' Problem and used that conception of steganographic channels to create an anonymized secure communications model that incorporated identity sets and channel state estimation.

We also realized that we needed a mechanism to assess the security properties of steganographic schemes. While seminal works like [28] and [29] provide some of the information theoretic and provable security bases for steganographic schemes, there was a lack of consideration for a Warden with more sophisticated abilities than simply monitoring an authorized communication channel. Thus, we introduced our ASAD framework to account for the Warden's ability to potentially forge, replay, reorder, or drop communications, borrowing from well-known AEAD concepts. To our knowledge this is the first work to extend classical AEAD concepts into the steganographic space.

7.1.2 Blockchain-based Embedding Algorithms

Unlike many specific embedding approaches in blockchain protocols (see Chapter 2), we presented an generic address embedding algorithm, ASGARD that had a much larger throughput than other provably secure address embedding schemes (see [20]). We used our ASAD framework to analyze our ASGARD construction and provided proofs regarding several of its properties including the Warden's ability to detect an embedded message (IND-CHA), as well as the Warden's ability to forge, replay, reorder, or drop transactions.

Finally, we presented a proof-of-concept practical implementation in Ethereum and performed a more thorough security analysis of the implementation using our ASAD framework and specific Ethereum implementation details. Importantly, we provided significant evidence that the output of the Keccak-256 hashing algorithm and the output of the AES-256 both appeared to be random and thus were indistinguishable from each other.

7.1.3 Analysis of Ethereum Blockchain Data

In the course of our work attempting to embed information in the Ethereum blockchain, we had to gain a detailed understanding of several of the underlying fields in an Ethereum transaction. In Chapter 5 and Chapter 6, we examined a number of those fields in Ethereum. In Chapter 5, we provided an analysis of the *value* field and its best-fit distribution over the last 12 months. We also used that information to determine about how much money would need to be sent per transaction as part of our practical scheme. We also used the *input data* field as an illustration of how making mistakes with associated data fields can easily tip the Warden off as to the presence of a suspicious transaction. In Chapter 6, we presented aggregate data metrics for malicious transaction detection, volumetric transaction rate and unique address activity. A combination of these metrics could enable the Warden to identify our embedded transactions, especially if we limited the size of the identity set. Thus, we were able to provide a set of recommendations, such as ensuring that an address only be used once in our scheme, to better prevent the Warden from identifying our communication efforts.

7.2 Future Work

Similar to our approach in Section 7.1, we also breakout future work opportunities by contribution. With regards to our steganographic extensions, we did not present a formal experiment to look at unlinkability. One avenue of future work would be presenting an experiment similar to the ones in our ASAD framework that encapsulated unlinkability. Furthermore, we also did not present a formalization for the digital ledger. Our work can be extended by incorporating security experiments related to the role of the ledger.

From the perspective of our blockchain-based embedding algorithm, our construction was only one out of several possible constructions. There is ample future work possible in presenting a different construction and analyzing it using our ASAD framework. In terms of practical implementation, exploiting the unique opportunities that specific blockchain protocols present (for example using Bitcoin instead of Ethereum, etc.), might provide another interesting avenue for extension. Furthermore, though we considered the consequences of address reuse in our ASGARD construction, there is ground to be covered in answering the question of how many times an address can be reused before requiring a switch to another address.

With regards to data modeling of the Ethereum blockchain, one area we did not explore was the presence of layer 2 or off-chain transactions where a number of actions can take place, but only the final result is stored on the blockchain ledger (see the Lightning network [140] as an example of off-chain transactions). As they become more common, these types of transactions could potentially alter the distribution of transaction traffic and the other metrics that we developed.

Another area in our exploration of data modeling of the Ethereum blockchain that we did not fully examine was using our metrics to detect malicious activity, whether via embedding or other nefarious activity. One interesting aspect is that both for volumetric transaction rate and for address analysis, the distributions are long-tailed. We believe that clustering around both the head and the long tail can provide important information as to normal network operation and potential outlier activity. This in turn could be used to identify malicious activity.

In conclusion, the proliferation of blockchain protocols and other privacy-based/anonymity-based protocols offers several unique and useful features for transmission of covert messages. Understanding those features, characterizing them, and leveraging them is critical to evolving the field of steganography.

APPENDIX A: Stateful AEAD Definition and Experiments

We present the Stateful Authenticated Encryption with Associated Data (AEAD) definition and security experiments from [27] as guides to our ASAD experiment and definition.

Definition **A.0.1.** *Let* Π *be a stateful AEAD scheme and let* $\mathcal{A}$ *be a PPT adversarial algorithm, let* $i \in \{1, \ldots, 4\}$ *and let* $b \in \{0, 1\}$. *The stateful AEAD experiment for* Π *with condition* cond_i *and bit* b *is given by* $\mathrm{Exp}_{\Pi}^{\mathsf{aead}_i - b}(\mathcal{A})$ *in Figure A.1. We define*

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

```
$\mathrm{Exp}^{\mathsf{auth_i}}_{\Pi,\mathcal{A}}()$:
1: $k \xleftarrow{\$} \mathrm{Kgn}()$
2: $st_E \leftarrow \perp, st_D \leftarrow \perp$
3: $u \leftarrow 0, v \leftarrow 0$
4: out-of-sync $\leftarrow 0$
5: $b' \xleftarrow{\$} \mathcal{A}^{\mathsf{Encrypt}(\cdot)\mathsf{Decrypt}(\cdot)}()$
6: return $b'$
```

```
Oracle Decrypt(ad, $c$)
1: if $b = 0$ then
2:     return $\perp$
3: end if
4: $v \leftarrow v + 1$
5: $rcvd.c_v \leftarrow c$
6: $(\mathsf{ad}, m, \alpha, st_D) \leftarrow \mathrm{D}(k, \mathsf{ad}, c, st_D)$
7: if $(i = 4) \land \mathsf{cond}_4$ then
8:     out-of-sync $\leftarrow 1$
9: else if $(\alpha = 1) \land \mathsf{cond}_i$ then
10:     out-of-sync $\leftarrow 1$
11: end if
12: if out-of-sync $= 1$ then
13:     return $m$
14: end if
15: return $\perp$
```

```
Oracle Encrypt($\ell$, ad, $m_0, m_1$)
1: $u \leftarrow u + 1$
2: $(sent.c^{(0)}, st_E^{(0)}) \leftarrow \mathrm{E}(k, \ell, \mathsf{ad}, m_0, st_E)$
3: $(sent.c^{(1)}, st_E^{(1)}) \leftarrow \mathrm{E}(k, \ell, \mathsf{ad}, m_1, st_E)$
4: if $sent.c^{(0)} = \perp$ or $sent.c^{(1)} = \perp$ then
5:     return $\perp$
6: end if
7: $(sent.ad_u, sent.c_u, st_E) := (\mathsf{ad}, sent.c^{(b)}, st_E^{(b)})$
8: return $sent.c_u$
```

Authentication Conditions

1. Basic authenticated stego:
   $\mathsf{cond}_1 = (\nexists w : (c = sent.c_w) \land (\mathsf{ad} = sent.ad_w))$
2. Basic authenticated stego, no replays:
   $\mathsf{cond}_2 = (\nexists w : (c = sent.c_w) \land (\mathsf{ad} = sent.ad_w)) \lor (\exists w < v : c = rcvd.c_w)$
3. Basic authenticated stego, no replays, strictly increasing:
   $\mathsf{cond}_3 = (\nexists w : (c = sent.c_w) \land (\mathsf{ad} = sent.ad_w)) \lor (\exists w, x, y : (w < v) \land (sent.c_x = rcvd.c_w) \land (sent.c_y = rcvd.c_v) \land (x \geq y))$
4. Basic authenticated stego, no replays, strictly increasing, no drops:
   $\mathsf{cond}_4 = (u < v) \lor (c \neq sent.c_v) \lor (\mathsf{ad} \neq sent.ad_v)$

Figure A.1. Stateful AEAD Experiment $\mathsf{auth_i}$ with Authentication Condition cond_i for Stateful AEAD Scheme Π = (Kgn, E, D) and Adversary $\mathcal{A}$

APPENDIX B:
NIST Randomness Test Output for Keccak-256 Hashes

This appendix presents the results of our comparison between the output of the Keccak-256 algorithm and random. As noted in Chapter 5, we used the NIST randomness tests [112] and the published code to perform this analysis.

```
------------------------------------------------------------------------------
RESULTS FOR THE UNIFORMITY OF P-VALUES AND THE PROPORTION OF PASSING SEQUENCES
------------------------------------------------------------------------------
   generator is <data/hedat_400mil>
------------------------------------------------------------------------------
 C1  C2  C3  C4  C5  C6  C7  C8  C9 C10  P-VALUE  PROPORTION  STATISTICAL TEST
------------------------------------------------------------------------------
 19  16  13  21  29  25  20  17  17  23  0.350485    198/200     Frequency
 19  14  21  29  14  22  18  23  25  15  0.268917    198/200     BlockFrequency
 21  16  14  20  20  19  24  22  20  24  0.875539    197/200     CumulativeSums
 19  13  19  29  22  14  18  22  27  17  0.219006    199/200     CumulativeSums
 23  15  22  13  25  21  24  22  19  16  0.585209    196/200     Runs
 19  14  25  15  22  27  22  18  20  18  0.574903    197/200     LongestRun
 19  26  18  23  25  12  17  18  15  27  0.255705    198/200     Rank
 33  22  18  15  19  21  14  17  14  27  0.053627    195/200     FFT
 14  23  20  24  19  18  21  18  24  19  0.883171    199/200     NonOverlappingTemplate
 23  21  22  15  19  14  33  21  15  17  0.122325    199/200     NonOverlappingTemplate
 20  24  15  16  28  21  15  25  20  16  0.401199    199/200     NonOverlappingTemplate
 24  21  22  16  15  15  22  24  22  19  0.779188    198/200     NonOverlappingTemplate
 26  20  17  17  15  24  20  25  22  14  0.534146    199/200     NonOverlappingTemplate
 29  18  18  16  23  15  19  20  20  22  0.616305    198/200     NonOverlappingTemplate
 22  21  28  18  17  18  24  15  19  18  0.678686    199/200     NonOverlappingTemplate
 18  23  24  26  19  15  15  13  27  20  0.296834    196/200     NonOverlappingTemplate
 17  18  27  17  17  27  27  18  17  15  0.289667    196/200     NonOverlappingTemplate
 26  12  25  27  14  18  21  22   9  26  0.026948    198/200     NonOverlappingTemplate
```

```
21  16  22  21  19  18  17  22  23  21  0.980883    199/200     NonOverlappingTemplate
20  28  32  13  19  21  18  14  16  19  0.071177    200/200     NonOverlappingTemplate
15  16  22  16  23  15  21  23  25  24  0.605916    197/200     NonOverlappingTemplate
17  24  12  18  20  31  14  25  21  18  0.122325    198/200     NonOverlappingTemplate
18  15  23  19  27  21  18  17  16  26  0.564639    198/200     NonOverlappingTemplate
13  17  17  17  30  19  14  27  25  21  0.108791    199/200     NonOverlappingTemplate
17  19  16  14  19  17  32  28  22  16  0.090936    198/200     NonOverlappingTemplate
19  27  22  19  22  17  18  19  16  21  0.875539    198/200     NonOverlappingTemplate
27  19  19  13  16  14  18  23  23  28  0.219006    195/200     NonOverlappingTemplate
19  18  25  22  14  11  23  26  25  17  0.242986    199/200     NonOverlappingTemplate
19  24  26  14  17  20  20  22  18  20  0.807412    196/200     NonOverlappingTemplate
26  19  21  16  21  16  21  24  19  17  0.842937    198/200     NonOverlappingTemplate
16  28  18  16  18  20  15  24  25  20  0.484646    199/200     NonOverlappingTemplate
24  14  21  15  17  18  19  25  23  24  0.626709    195/200     NonOverlappingTemplate
19  16  14  20  22  22  21  17  26  23  0.759756    200/200     NonOverlappingTemplate
21  21  24  17  23  17  15  20  19  23  0.911413    197/200     NonOverlappingTemplate
21  26  20  24  13  20  18  18  19  21  0.779188    199/200     NonOverlappingTemplate
23  21  22  21  16  19  14  22  17  25  0.807412    199/200     NonOverlappingTemplate
19  17  24  21  18  23  14  11  34  19  0.038818    197/200     NonOverlappingTemplate
21  20  13  18  17  26  23  21  21  20  0.788728    194/200     NonOverlappingTemplate
23  13  27  12  14  16  23  20  26  26  0.085587    200/200     NonOverlappingTemplate
20  24  19  19  16  18  12  24  30  18  0.268917    196/200     NonOverlappingTemplate
23  14  26  17  27  17  17  18  16  25  0.342451    199/200     NonOverlappingTemplate
18  24  22  20  20  15  15  21  27  18  0.699313    200/200     NonOverlappingTemplate
21  16  15  22  15  26  25  17  21  22  0.605916    196/200     NonOverlappingTemplate
20  18  21  14  16  21  19  22  30  19  0.514124    198/200     NonOverlappingTemplate
16  29  26  20  20  11  24  18  23  13  0.102526    197/200     NonOverlappingTemplate
23  13  28  19  19  30  14  12  22  20  0.058984    199/200     NonOverlappingTemplate
15  26  26  25  11  17  20  18  23  19  0.255705    198/200     NonOverlappingTemplate
25  27  12  17  23  20  19  15  17  25  0.289667    199/200     NonOverlappingTemplate
26  34  18  18  20  15  24  16  15  14  0.036352    199/200     NonOverlappingTemplate
28  21  20  18  23  12  24  22  19  13  0.304126    198/200     NonOverlappingTemplate
26  21  18  16  27  17  18  16  15  26  0.366918    200/200     NonOverlappingTemplate
18  19  25  23  16  22  21  14  19  23  0.807412    198/200     NonOverlappingTemplate
13   8  27  22  21  26  21  16  21  25  0.060875    198/200     NonOverlappingTemplate
```

```
17  20  15  23  18  23  21  24  18  21  0.917870    198/200     NonOverlappingTemplate
17  16  26  19  25  18  12  16  24  27  0.224821    199/200     NonOverlappingTemplate
21  16  17  32  18  25  16  16  24  15  0.137282    199/200     NonOverlappingTemplate
17  23  24  24  21  15  24  15  16  21  0.668321    199/200     NonOverlappingTemplate
14  23  22  17  18  21  18  25  22  20  0.851383    200/200     NonOverlappingTemplate
20  20  24  20  17  22  15  17  18  27  0.759756    199/200     NonOverlappingTemplate
24  23  20  23  12  17  15  22  24  20  0.574903    198/200     NonOverlappingTemplate
22  16  26  14  17  25  23  15  24  18  0.437274    196/200     NonOverlappingTemplate
18  22  16  16  10  24  21  23  26  24  0.282626    196/200     NonOverlappingTemplate
14  10  23  27  23  18  20  18  27  20  0.162606    200/200     NonOverlappingTemplate
19  20  19  24  24  14  25  17  18  20  0.798139    197/200     NonOverlappingTemplate
19  18  24  17  22  12  22  30  18  18  0.311542    200/200     NonOverlappingTemplate
25  18  19  19  19  20  14  20  21  25  0.859637    199/200     NonOverlappingTemplate
24  21  24  24  21  13  13  19  23  18  0.524101    199/200     NonOverlappingTemplate
20  18  18  19  22  14  19  20  24  26  0.825505    200/200     NonOverlappingTemplate
24  17  15  20  13  26  24  23  19  19  0.524101    198/200     NonOverlappingTemplate
25  17  18  19  19  16  18  14  23  31  0.255705    200/200     NonOverlappingTemplate
18  14  20  21  23  21  26  22  13  22  0.616305    197/200     NonOverlappingTemplate
28  19  25  16  25  12  23  22  18  12  0.129620    197/200     NonOverlappingTemplate
20  18  17  24  17  23  19  24  16  22  0.897763    197/200     NonOverlappingTemplate
25  14  17  19  16  22  20  25  21  21  0.749884    196/200     NonOverlappingTemplate
20  26  23  14  22  15  21  21  22  16  0.678686    198/200     NonOverlappingTemplate
26  19  13  18  26  28  18  13  17  22  0.171867    195/200     NonOverlappingTemplate
18  21  14  31  25  19  20  13  20  19  0.219006    199/200     NonOverlappingTemplate
24  23  26  18  13  25  18  22  20  11  0.249284    199/200     NonOverlappingTemplate
24  20  18  16  13  27  13  27  15  27  0.083018    196/200     NonOverlappingTemplate
19  25  12  26  19  19  22  15  27  16  0.268917    196/200     NonOverlappingTemplate
19  16  22  21  24  21  18  19  17  23  0.960198    197/200     NonOverlappingTemplate
21  24  19  17  22  19  16  25  25  12  0.524101    197/200     NonOverlappingTemplate
14  23  20  24  19  18  21  18  24  19  0.883171    199/200     NonOverlappingTemplate
23  19  18  19  22  27  16  16  21  19  0.825505    197/200     NonOverlappingTemplate
12  26  22  23  16  22  18  22  15  24  0.428095    198/200     NonOverlappingTemplate
17  22  17  15  20  28  19  21  23  18  0.709558    200/200     NonOverlappingTemplate
14  25  18  19  21  17  22  27  14  23  0.465415    198/200     NonOverlappingTemplate
14  24  22  19  25  23  22  15  17  19  0.689019    197/200     NonOverlappingTemplate
```

```
26  16  34  12  15  16  22  15  24  20  0.018540    199/200      NonOverlappingTemplate
20  23  23  25  14  20  18  19  21  17  0.859637    195/200      NonOverlappingTemplate
21  16  19  22  21  28  18  18  20  17  0.816537    198/200      NonOverlappingTemplate
19  14  26  20  30  15  15  22  20  19  0.249284    200/200      NonOverlappingTemplate
16  22  11  17  28  17  23  19  23  24  0.282626    200/200      NonOverlappingTemplate
19  21  15  26  14  26  23  14  22  20  0.419021    198/200      NonOverlappingTemplate
22  18  23  20  19  18  18  18  23  21  0.991468    198/200      NonOverlappingTemplate
19  16  17  21  25  24  29  19  12  18  0.282626    200/200      NonOverlappingTemplate
16  20  23  16  19  24  16  24  19  23  0.834308    200/200      NonOverlappingTemplate
21  20  24  24  23  18  22  15  19  14  0.779188    197/200      NonOverlappingTemplate
23  21  18  22  19  24   8  27  14  24  0.122325    196/200      NonOverlappingTemplate
19  31  25  17  14  14  25  17  22  16  0.118812    200/200      NonOverlappingTemplate
22  22  14  17  23  17  15  14  31  25  0.125927    199/200      NonOverlappingTemplate
22  17  19  21  19  15  15  28  20  24  0.605916    195/200      NonOverlappingTemplate
18  15  16  25  26  15  18  26  20  21  0.474986    200/200      NonOverlappingTemplate
26  15  18  23  15  13  17  26  22  25  0.268917    198/200      NonOverlappingTemplate
15  25  22  17  15  21  24  24  22  15  0.585209    198/200      NonOverlappingTemplate
12  22  23  25  22  24   9  27  22  14  0.055361    199/200      NonOverlappingTemplate
28  21  16  18  20  16  21  14  26  20  0.465415    194/200      NonOverlappingTemplate
18  22  20  24  18  26  22  12  18  20  0.657933    199/200      NonOverlappingTemplate
22  18  20  17  30  21  19   8  20  25  0.108791    195/200      NonOverlappingTemplate
19  11  24  18  16  22  25  23  19  23  0.504219    199/200      NonOverlappingTemplate
18  27  21  23  24  13  21  21  23   9  0.162606    199/200      NonOverlappingTemplate
14  25  18  16  24  23  30  15  18  17  0.202268    200/200      NonOverlappingTemplate
22  21  15  19  20  11  21  23  25  23  0.554420    196/200      NonOverlappingTemplate
14  19  17  20  28  25  19  24  13  21  0.342451    199/200      NonOverlappingTemplate
20  17  19  22  16  17  21  16  22  30  0.534146    199/200      NonOverlappingTemplate
20  17  23  25  23  14  20  18  22  18  0.834308    197/200      NonOverlappingTemplate
19  15  23  22  16  19  20  22  17  27  0.749884    200/200      NonOverlappingTemplate
19  27  13  23  22  18  19   7  22  30  0.021262    200/200      NonOverlappingTemplate
20  23  24  17  17  25  13  29  14  18  0.219006    198/200      NonOverlappingTemplate
13  20  20  24  21  22  22  23  21  14  0.739918    198/200      NonOverlappingTemplate
17  21  22  19  26  27  15  17  20  16  0.585209    199/200      NonOverlappingTemplate
13  22  28  18  25  19  24  18  19  14  0.334538    196/200      NonOverlappingTemplate
11  29  23  18  25  26  20  16  12  20  0.071177    197/200      NonOverlappingTemplate
```

```
14  15  20  20  19  18  23  32  27  12  0.055361    197/200     NonOverlappingTemplate
16  23  25  22  19  14  31  21  16  13  0.125927    199/200     NonOverlappingTemplate
19  15  21  23  21  18  21  19  23  20  0.978072    199/200     NonOverlappingTemplate
23  23  18  22  18  14  22  22  22  16  0.859637    200/200     NonOverlappingTemplate
26  23  18  18  20  18  17  18  20  22  0.930026    198/200     NonOverlappingTemplate
21  22  22  21  21  12  19  30  11  21  0.167184    200/200     NonOverlappingTemplate
16  17  31  24  23  14  22  15  19  19  0.219006    199/200     NonOverlappingTemplate
19  17  24  16  20  19  14  29  18  24  0.437274    198/200     NonOverlappingTemplate
19  19  20  21  23  19  27  22  12  18  0.668321    196/200     NonOverlappingTemplate
19  15  21  21  24  23  19  24  17  17  0.883171    199/200     NonOverlappingTemplate
25  18  17  15  19  19  20  18  23  26  0.769527    199/200     NonOverlappingTemplate
23  17  17  16  31  20  18  14  26  18  0.202268    198/200     NonOverlappingTemplate
21  23  24  23  15  18  14  21  18  23  0.769527    199/200     NonOverlappingTemplate
18  14  26  20  15  31  19  11  22  24  0.062821    199/200     NonOverlappingTemplate
16  15  18  17  21  19  29  25  18  22  0.484646    197/200     NonOverlappingTemplate
19  16  18  20  24  15  17  22  26  23  0.739918    197/200     NonOverlappingTemplate
22  26  15  18  14  20  10  24  21  30  0.064822    198/200     NonOverlappingTemplate
23  20  22  25  20  24  19  14  24   9  0.249284    200/200     NonOverlappingTemplate
20  23  18  25  22  16  23  22  18  13  0.719747    197/200     NonOverlappingTemplate
20  21  17  19  19  21  19  20  24  20  0.997147    198/200     NonOverlappingTemplate
19  15  20  14  22  24  26  23  21  16  0.616305    199/200     NonOverlappingTemplate
17  21  19  22  24  19  29  22  19   8  0.158133    198/200     NonOverlappingTemplate
19  22  27  25  19  19  14  19  20  16  0.668321    198/200     NonOverlappingTemplate
17  18  13  22  22  20  14  26  21  27  0.383827    198/200     NonOverlappingTemplate
13  19  23  16  27  26  20  30  13  13  0.036352    197/200     NonOverlappingTemplate
19  20  20  23  17  20  20  22  24  15  0.955835    199/200     NonOverlappingTemplate
18  19  23  20  15  21  16  22  24  22  0.911413    198/200     NonOverlappingTemplate
24  15  17  22  22  16  18  21  18  27  0.678686    198/200     NonOverlappingTemplate
19  22  21  15  21  18  22  18  17  27  0.825505    198/200     NonOverlappingTemplate
18  17  24  17  26  18  19  21  16  24  0.779188    198/200     NonOverlappingTemplate
19  20  24  15  23  21  15  20  22  21  0.904708    198/200     NonOverlappingTemplate
19  19  22  19  19  25  17  20  23  17  0.964295    197/200     NonOverlappingTemplate
21  24  19  16  23  19  15  26  25  12  0.375313    197/200     NonOverlappingTemplate
27  20  24  18  21  11  26  18  19  16  0.319084    198/200     OverlappingTemplate
18  14  24  24  18  16  26  17  21  22  0.626709    198/200     Universal
```

```
 26  18  25  21  13  19  19  17  24  18  0.605916    198/200     ApproximateEntropy
 17  12  19  11   9  15  15  12  23  10  0.186566    142/143     RandomExcursions
 15  12  14  14  16  13  16  13  14  16  0.998074    141/143     RandomExcursions
 16  11  11  19  14  14  15  15  13  15  0.933310    140/143     RandomExcursions
 15  14  21  11  15  11  16  12  15  13  0.775062    142/143     RandomExcursions
 10  21  20  12  12  14  13  12  13  16  0.484646    142/143     RandomExcursions
 11  13  15  11  21  13   9  20  15  15  0.392456    143/143     RandomExcursions
 15  10  13  14  19  13  23  14   9  13  0.290684    142/143     RandomExcursions
 12  15  12  12  22  12  16  13  13  16  0.689019    141/143     RandomExcursions
 14  14  12  13  12  16  17  17  16  12  0.972188    141/143     RandomExcursionsVariant
 14  12  17  13  12  18  13  16  13  15  0.967060    141/143     RandomExcursionsVariant
 14  12  21  11  15  16  17  13   8  16  0.498594    141/143     RandomExcursionsVariant
 15  16  17  14   7  17  10  16  14  17  0.614820    141/143     RandomExcursionsVariant
 17  17   9  14  15  14  16  11  20  10  0.555877    141/143     RandomExcursionsVariant
 14  14  16  19  17  13   4  16  17  13  0.280637    141/143     RandomExcursionsVariant
 17   8  21  15  19  11  17  15   8  12  0.172545    141/143     RandomExcursionsVariant
 12  16  14  16  10  16  17  16  11  15  0.916051    141/143     RandomExcursionsVariant
  9  13  19  17  16   7  17  18  12  15  0.333418    143/143     RandomExcursionsVariant
 12  17  11   9  14  14  11  12  15  28  0.031249    143/143     RandomExcursionsVariant
 18  10  11   9   7  20   7  19  17  25  0.002600    143/143     RandomExcursionsVariant
 16   9  14   9  22   9  12  16  20  16  0.135609    143/143     RandomExcursionsVariant
 12  20   6  16  16   9  13  24  15  12  0.043408    142/143     RandomExcursionsVariant
 14  11  17  12  11  17  20  12  15  14  0.761159    142/143     RandomExcursionsVariant
 14  13  12   4  23  21  14  17  13  12  0.037743    143/143     RandomExcursionsVariant
 13  11  12  14  17  18  17  12  16  13  0.906649    143/143     RandomExcursionsVariant
 10  13  14  14  16  15  16  13  16  16  0.980883    143/143     RandomExcursionsVariant
 11   9  16  18  18  10   9  20  19  13  0.193916    143/143     RandomExcursionsVariant
 17  28  19  29  19  27  19  16  11  15  0.058984    199/200     Serial
 29  19  21  23  20  16  23  18  20  11  0.342451    196/200     Serial
 12  24  26  17  20  22  24  16  21  18  0.504219    198/200     LinearComplexity


- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
The minimum pass rate for each statistical test with the exception of the
random excursion (variant) test is approximately = 193 for a
```

```
sample size = 200 binary sequences.

The minimum pass rate for the random excursion (variant) test
is approximately = 138 for a sample size = 143 binary sequences.

For further guidelines construct a probability table using the MAPLE program
provided in the addendum section of the documentation.
- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
```

THIS PAGE INTENTIONALLY LEFT BLANK

APPENDIX C:
Code

This appendix presents a number of useful resources for researchers attempting to replicate or extend our work. Figure C.1 shows a breakdown of how this appendix will be structured.

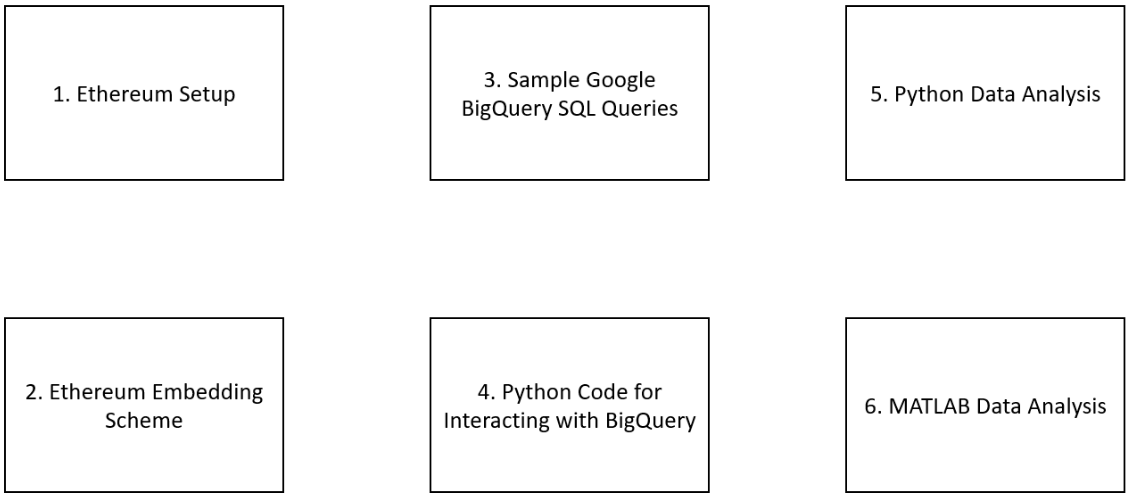
1. Ethereum Setup
3. Sample Google BigQuery SQL Queries
5. Python Data Analysis
2. Ethereum Embedding Scheme
4. Python Code for Interacting with BigQuery
6. MATLAB Data Analysis

Figure C.1. Code Repository Appendix Structure

C.1 Ethereum Setup

Please see our work in [141] for a detailed guide for setting up an Ethereum node. Note that some commands will have changed as the APIs change frequently.

C.2 Ethereum Embedding Scheme

The idea behind the Ethereum embedding scheme is relatively straightforward. We encrypt a target message with an encryption algorithm of our choice and use the resultant ciphertext as the receiver address for an Ethereum transaction. On the decoding and decryption side, we check each block added to our local Ethereum blockchain, ensure that each transaction comes from an authorized sender account, and extract/decrypt the message. We used Python3 as our language of choice and the following libraries for this function.

Listing C.1: Libraries used for Python

```
from web3 import Web3
import random
import hashlib
from base64 import b64decode
from base64 import b64encode
import json
from Crypto.Cipher import AES
from Crypto.Random import get_random_bytes
from Crypto.Util.Padding import pad, unpad
```

C.3 Sample Google BigQuery SQL Queries

In this section, an example of a SQL query that selects data from the Google BigQuery database for Ethereum, specifically *bigquery-public-data.crypto_ethereum.transactions* and *bigquery-public-data.crypto_ethereum.contracts*.

Number of Transactions in a Day

```
select count(*) from `bigquery-public-data.crypto_ethereum.transactions`
where `block_timestamp` > '2021-02-15 00:00:00 UTC' and `block_timestamp`
< '2021-02-15 23:59:59 UTC'
```

C.4 Python Code for Interacting with BigQuery

In this section, we discuss considerations for Python scripts that construct SQL queries and then create data files with our desired formats. We emphasize that one must first register an account with Google BigQuery and get an API key in order to use these resources. Detailed guides on this process can be found at [142] and [143]. Also, one must make sure to export the credential location to $PATH (something like export GOOGLE_APPLICATION_CREDENTIALS="/home/path_to/creds.json) before importing bigquery. The following code listing is an example of using BigQuery in a Python Script.

Listing C.2: Python Setup Example for Google BigQuery

```
from google.cloud import bigquery
from datetime import datetime, date, time, timezone, timedelta
import matplotlib.pyplot as plt
import pandas as pd

client = bigquery.Client()

sql = """
select count(*) from `bigquery-public-data.crypto_ethereum.transactions`
where `block_timestamp` > '2021-02-15 00:00:00 UTC' and `block_timestamp`<
'2021-02-15 23:59:59 UTC'
"""
```

C.5 Data Analysis using Python and MATLAB

The data analysis that was performed in Chapter 5 and Chapter 6 used a combination of Python3 and MATLAB. We used the following libraries in Python3.

Listing C.3: Python Script for Curve Fitting

```
import matplotlib.pyplot as plt
import pandas as pd
from scipy.optimize import curve_fit
import scipy.stats
import numpy as np
from lmfit.models import PowerLawModel, LinearModel,
ExponentialGaussianModel, GaussianModel, LognormalModel
```

For MATLAB, we used primarily built-in statistics functions. We also made significant use of the *fitmethis* package [83] for our curve fitting process.

THIS PAGE INTENTIONALLY LEFT BLANK

List of References

[1] C. Timberg, "The real story of how the Internet became so vulnerable," The Washington Post, May. 30, 2015 [Online]. Available: http://www.washingtonpost.com/sf/business/2015/05/30/net-of-insecurity-part-1/

[2] V. Kanth, C. Bollmann, M. Tummala, and J. C. McEachen, "A novel adaptable framework for covert communications in anonymized protocols," in *MILCOM 2021 - 2021 IEEE Military Communications Conference (MILCOM)*, 2021, pp. 451–457.

[3] V. Kanth, J. McEachen, and M. Tummala, "Parameter identification for malicious transaction detection in blockchain protocols," in *Blockchain and Applications*, J. Prieto, A. Partida, P. Leitão, and A. Pinto, Eds. Cham: Springer International Publishing, 2022, pp. 54–63.

[4] L. Rainie, Janna, and Erson, "Theme 6: Trust will diminish because the internet is not secure and powerful forces threaten individuals' rights," Pew Research Center, Aug. 10, 2017 [Online]. Available: https://www.pewresearch.org/internet/2017/08/10/theme-6-trust-will-diminish-because-the-internet-is-not-secure-and-powerful-forces-threaten-individuals-rights/

[5] K. A. Wallace, "Anonymity," *Ethics and Information Technology*, vol. 1, no. 1, pp. 21–31, Mar. 1999 [Online]. Available: http://link.springer.com/10.1023/A:1010066509278

[6] J. Goodchild, "Cybercrime's most lucrative careers," Dark Reading, Dec. 31, 2019 [Online]. Available: https://www.darkreading.com/edge-articles/cybercrime-s-most-lucrative-careers

[7] R. Lishchuk, "Most desired data: Whose is the most in demand, and how much is it worth?" Mackeeper, Nov. 16 2020 [Online]. Available: https://mackeeper.com/blog/most-desired-data/

[8] R. S. Mueller III, "Report on the investigation into Russian interference in the 2016 presidential election. volumes i & ii," Department of Justice, Mar. 2019 [Online]. Available: https://www.justice.gov/archives/sco/file/1373816/download

[9] D. O'Sullivan, D. Griffin, and C. Devine, "In attempt to sow fear, Russian trolls paid for self-defense classes for African Americans," CNN, Oct. 18, 2017 [Online]. Available: https://money.cnn.com/2017/10/18/media/black-fist-russia-self-defense-classes/index.html

[10] A. O'Driscoll, "Does your VPN keep logs? 140 VPN logging policies revealed," Comparitech, Mar. 10, 2021 [Online]. Available: https://www.comparitech.com/vpn/vpn-logging-policies/

[11] S. Frenkel, "This is how ISIS uses the Internet," BuzzFeed News, May. 12, 2016 [Online]. Available: https://www.buzzfeednews.com/article/sheerafrenkel/everything-you-ever-wanted-to-know-about-how-isis-uses-the-i

[12] Department of Justice, "Global disruption of three terror finance cyber-enabled campaigns," Aug 13, 2020 [Online]. Available: https://www.justice.gov/opa/pr/global-disruption-three-terror-finance-cyber-enabled-campaigns

[13] J. Leyden, "The 'one tiny slip' that put LulzSec chief Sabu in the FBI's pocket," The Register, Mar. 7, 2012 [Online]. Available: https://www.theregister.com/2012/03/07/lulzsec_takedown_analysis/

[14] K. Cabaj, L. Caviglione, W. Mazurczyk, S. Wendzel, A. Woodward, and S. Zander, "The new threats of information hiding: The road ahead," *IT Professional*, vol. 20, no. 3, pp. 31–39, 2018 [Online].

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

[16] Z. Wu, *Information hiding in speech signals for secure communication*. Rockland, MA, USA: Syngress, 2015.

[17] W. Mazurczyk, S. Wendzel, S. Zander, A. Houmansadr, and K. Szczypiorski, Eds., *Information Hiding in Communication Networks: Fundamentals, Mechanisms, Applications, and Countermeasures*. Hoboken, New Jersey: John Wiley & Sons, Inc., Feb. 2016. Available: http://doi.wiley.com/10.1002/9781119081715

[18] B. Lampson, "A note on the confinement problem," *Communications of the ACM*, vol. 16, no. 10, pp. 613–615, 1973.

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

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

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

[22] K. Bock, iyouport, Anonymous, L.-H. Merino, D. Fifield, A. Houmansadr, and D. Levin, “Exposing and circumventing China’s censorship of ESNI,” censorship.ai, Aug. 7, 2020 [Online]. Available: https://geneva.cs.umd.edu/posts/china-censors-esni/esni/

[23] L. Kuo, “China appears to block Signal app, tightening Internet controls,” Washington Post, Mar. 16, 2021 [Online]. Available: https://www.washingtonpost.com/world/asia_pacific/signal-app-china-blocked-firewall/2021/03/16/a7972b52-860c-11eb-be4a-24b89f616f2c_story.html

[24] K. Bradsher, “China blocks WhatsApp, broadening online censorship,” The New York Times, Sep. 25, 2017 [Online]. Available: https://www.nytimes.com/2017/09/25/business/china-whatsapp-blocked.html

[25] L. Whitney, “Search engine DuckDuckGo blocked in China,” CNET, Sep. 22 2014 [Online]. Available: https://www.cnet.com/tech/services-and-software/search-engine-duckduckgo-now-blocked-in-china/

[26] P. Rogaway, “Authenticated-encryption with associated-data,” in *Proceedings of the 9th ACM Conference on Computer and Communications Security* (CCS ’02). New York, NY, USA: Association for Computing Machinery, 2002 [Online], p. 98–107. Available: https://doi.org/10.1145/586110.586125

[27] 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 [Online], https://ia.cr/2015/1150.

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

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

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

[31] A. Pfitzmann and M. Hansen, “A terminology for talking about privacy by data minimization: Anonymity, unlinkability, undetectability, unobservability, pseudonymity, and identity management,” IETF, Aug. 10 2010, [Online]. Available: http://dud.inf.tu-dresden.de/literatur/Anon_Terminology_v0.34.pdf

[32] ISO Central Secretary, “Information technology – security techniques – evaluation criteria for it security,” International Organization for Standardization, Geneva, CHE, Standard ISO/IEC 15408-2, 2008 [Online]. Available: https://www.iso.org/standard/72892.html

[33] C. E. Shannon, “Communication theory of secrecy systems,” *The Bell System Technical Journal*, vol. 28, no. 4, pp. 656–715, 1949.

[34] D. Chaum, “Security without identification: Transaction systems to make big brother obsolete,” *Commun. ACM*, vol. 28, no. 10, p. 1030–1044, oct 1985 [Online]. Available: https://doi.org/10.1145/4372.4373

[35] Monero, “The Monero project,” Accessed Sep. 6, 2020 [Online]. Available: https://getmonero.org/index.html

[36] 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.

[37] J. Katz and M. Yung, “Unforgeable encryption and chosen ciphertext secure modes of operation,” in *Fast Software Encryption, 7th International Workshop, FSE 2000, New York, NY, USA, April 10-12, 2000, Proceedings* (Lecture Notes in Computer Science). Springer, 2000, vol. 1978, pp. 284–299.

[38] M. Bellare, P. Rogaway, and D. Wagner, “EAX: A conventional authenticated-encryption mode,” Cryptology ePrint Archive, Report 2003/069, 2003 [Online], https://ia.cr/2003/069.

[39] M. Bellare, T. Kohno, and C. Namprempre, “Authenticated encryption in SSH: Provably fixing the SSH binary packet protocol,” in *Proceedings of the 9th ACM Conference on Computer and Communications Security* (CCS ’02). New York, NY, USA: Association for Computing Machinery, 2002 [Online], p. 1–11. Available: https://doi.org/10.1145/586110.586112

[40] T. Kohno, A. Palacio, and J. Black, “Building secure cryptographic transforms, or how to encrypt and MAC,” Cryptology ePrint Archive, Report 2003/177, 2003 [Online], https://ia.cr/2003/177.

[41] 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.

[42] M. Bauer, “New covert channels in HTTP: Adding unwitting web browsers to anonymity sets,” in *Proceedings of the 2003 ACM Workshop on Privacy in the Electronic Society* (WPES ’03). New York, NY, USA: Association for Computing Machinery, 2003 [Online], p. 72–78. Available: https://doi-org.libproxy.nps.edu/10.1145/1005140.1005152

[43] A. W. Dent, “Fundamental problems in provable security and cryptography,” *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences*, vol. 364, no. 1849, pp. 3215–3230, 2006.

[44] W. Diffie and M. Hellman, “New directions in cryptography,” *IEEE Transactions on Information Theory*, vol. 22, no. 6, pp. 644–654, 1976.

[45] M. O. Rabin, “Digitalized signatures and public-key functions as intractable as factorization,” Massachusetts Institute of Technology, Boston, MA, USA, Tech. Rep., 1979 [Online]. Available: https://dl.acm.org/doi/book/10.5555/889813

[46] D. Stebila, “An introduction to provable security,” Queensland University of Technology, Queensland, Australia, Tech. Rep., July 2014 [Online]. Available: http://files.douglas.stebila.ca/files/teaching/amsi-winter-school/Lecture-2-3-Provable-security.pdf

[47] M. Bellare and P. Rogaway, “Random oracles are practical: A paradigm for designing efficient protocols,” in *Proceedings of the 1st ACM Conference on Computer and Communications Security* (CCS ’93). New York, NY, USA: Association for Computing Machinery, 1993 [Online], p. 62–73. Available: https://doi.org/10.1145/168588.168596

[48] V. Shoup, “Lower bounds for discrete logarithms and related problems,” in *Advances in Cryptology — EUROCRYPT ’97*, W. Fumy, Ed. Berlin, Heidelberg: Springer Berlin Heidelberg, 1997, pp. 256–266.

[49] M. Rosulek. (2021, Jan. 3,). *The Joy of Cryptography*. [Online]. Available: https://joyofcryptography.com

[50] S. Goldwasser and S. Micali, “Probabilistic encryption,” *Journal of Computer and System Sciences*, vol. 28, no. 2, pp. 270–299, 1984 [Online]. Available: https://www.sciencedirect.com/science/article/pii/0022000084900709

[51] 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.

[52] S. Nakamoto, “Bitcoin: A peer-to-peer electronic cash system,” Bitcoin, 2009 [Online]. Available: https://bitcoin.org/bitcoin.pdf

[53] D. Di Francesco Maesa and P. Mori, “Blockchain 3.0 applications survey,” *Journal of Parallel and Distributed Computing*, vol. 138, pp. 99–114, 2020 [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0743731519308664

[54] S. Haber and W. S. Stornetta, “How to time-stamp a digital document,” *Journal of Cryptology*, vol. 3, no. 2, pp. 99–111, Jan. 1991 [Online]. Available: http://link.springer.com/10.1007/BF00196791

[55] I. B. Damgård, “Collision free hash functions and public key signature schemes,” in *Advances in Cryptology — EUROCRYPT’ 87*, D. Chaum and W. L. Price, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 1988, pp. 203–216.

[56] A. Rosic, “What is hashing? Step-by-Step guide-under hood Of blockchain,” Blockgeeks, May. 4, 2020 [Online]. Available: https://blockgeeks.com/guides/what-is-hashing/

[57] Ethereum, “Ethereum,” Accessed Jul. 19, 2019 [Online]. Available: https://ethereum.org

[58] A. Azaria, A. Ekblaw, T. Vieira, and A. Lippman, “Medrec: Using blockchain for medical data access and permission management,” in *2016 2nd International Conference on Open and Big Data (OBD)*, Aug 2016, pp. 25–30.

[59] S. Radocchia, “3 innovative ways blockchain will build trust in the food industry,” Forbes, Apr. 26, 2019 [Online]. Available: https://www.forbes.com/sites/samantharadocchia/2018/04/26/3-innovative-ways-blockchain-will-build-trust-in-the-food-industry/

[60] N. Alexopoulos, E. Vasilomanolakis, N. R. Ivánkó, and M. Mühlhäuser, “Towards blockchain-based collaborative intrusion detection systems,” in *Critical Information Infrastructures Security*, G. D’Agostino and A. Scala, Eds. Cham: Springer International Publishing, 2018, pp. 107–118.

[61] V. Kanth, A. McAbee, M. Tummala, and J. McEachen, “Collaborative intrusion detection leveraging blockchain and pluggable authentication modules,” in *Hawaii International Conference on System Sciences 2020*, 2020 [Online]. Available: https://hdl.handle.net/10125/64564

[62] T. K. Sharma, “Permissioned and permissionless blockchains: A comprehensive guide,” Blockchain Council, Nov. 2019 [Online]. Available: https://www.blockchain-council.org/blockchain/permissioned-and-permissionless-blockchains-a-comprehensive-guide/

[63] IBM, “What is blockchain technology? - IBM blockchain,” Accessed Mar. 29, 2022 [Online]. Available: https://www.ibm.com/topics/what-is-blockchain

[64] L. Lamport, R. Shostak, and M. Pease, “The Byzantine Generals Problem,” *ACM Transactions on Programming Languages and Systems (TOPLAS)*, vol. 4, no. 3, pp. 382–401, July 1982 [Online]. Available: http://dl.acm.org/doi/10.1145/357172.357176

[65] A. Gervais, G. O. Karame, K. Wüst, V. Glykantzis, H. Ritzdorf, and S. Capkun, “On the security and performance of proof of work blockchains,” in *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security* (CCS ’16). New York, NY, USA: Association for Computing Machinery, 2016 [Online], p. 3–16. Available: https://doi.org/10.1145/2976749.2978341

[66] W. Li, S. Andreina, J.-M. Bohli, and G. Karame, “Securing proof-of-stake blockchain protocols,” in *Data Privacy Management, Cryptocurrencies and Blockchain Technology*, J. Garcia-Alfaro, G. Navarro-Arribas, H. Hartenstein, and J. Herrera-Joancomartí, Eds. Cham: Springer International Publishing, 2017, pp. 297–315.

[67] M. Castro, B. Liskov *et al.*, “Practical byzantine fault tolerance,” in *OSDI*, no. 1999, 1999, vol. 99, pp. 173–186.

[68] L. Bach, B. Mihaljevic, and M. Zagar, “Comparative analysis of blockchain consensus algorithms,” in *2018 41st International Convention on Information and Communication Technology, Electronics and Microelectronics (MIPRO)*. IEEE, 2018, pp. 1545–1550.

[69] Ethereum, “Transactions,” Accessed Jul. 16, 2021 [Online]. Available: https://ethereum.org/en/developers/docs/transactions/

[70] Ethereum, “Ethereum accounts,” Accessed Nov. 29, 2021 [Online]. Available: https://ethereum.org/en/developers/docs/accounts/

[71] G. Wood, “Ethereum yellow paper,” Ethereum, Oct. 30, 2018 [Online]. Available: https://github.com/ethereum/yellowpaper

[72] Ethereum, “Consensus mechanisms,” Accessed Mar. 29, 2021 [Online]. Available: https://ethereum.org

[73] Ethereum Wiki, “dagger-hashimoto,” Accessed Mar. 29, 2022 [Online]. Available: https://eth.wiki/concepts/dagger-hashimoto

[74] M. Bedford Taylor, “The evolution of bitcoin hardware,” *Computer*, vol. 50, no. 9, pp. 58–66, 2017.

[75] Ethereum Wiki, “ethash,” Accessed Mar. 29, 2022 [Online]. Available: https://eth.wiki/en/concepts/ethash/ethash

[76] minerstat, “DAG size calculator and calendar,” Accessed Mar. 29, 2022 [Online]. Available: https://minerstat.com/dag-size-calculator

[77] Bit2Me Academy, “What is the Ethash mining algorithm?” Accessed Mar. 29, 2022 [Online]. Available: https://academy.bit2me.com/en/what-is-the-algorithm-of-ethash-mining/

[78] A. Fionov, “Exploring covert channels in Bitcoin transactions,” in *2019 International Multi-Conference on Engineering, Computer and Information Sciences (SIBIRCON)*, 2019, pp. 0059–0064.

[79] R. Matzutt, J. Hiller, M. Henze, J. H. Ziegeldorf, D. Müllmann, O. Hohlfeld, and K. Wehrle, “A quantitative analysis of the impact of arbitrary blockchain content on bitcoin,” in *Financial Cryptography and Data Security*, S. Meiklejohn and K. Sako, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2018, pp. 420–438.

[80] 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.

[81] 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 [Online], https://ia.cr/2021/1073.

[82] S. Liu, Z. Fang, F. Gao, B. Koussainov, Z. Zhang, J. Liu, and L. Zhu, “Whispers on Ethereum: Blockchain-based covert data embedding schemes,” in *Proceedings of the 2nd ACM International Symposium on Blockchain and Secure Critical Infrastructure* (BSCI ’20). New York, NY, USA: Association for Computing Machinery, 2020 [Online], p. 171–179. Available: https://doi.org/10.1145/3384943.3409433

[83] F. de Castro, “fitmethis,” MathWorks, Oct. 15, 2021 [Online]. Available: https://www.mathworks.com/matlabcentral/fileexchange/40167-fitmethis

[84] PennState: Statistics Online Courses, “1.2 - Maximum Likelihood Estimation,” Accessed Apr. 3, 2022 [Online]. Available: https://online.stat.psu.edu/stat415/lesson/1/1.2

[85] C. Bollmann, “Network anomaly detection with stable distributions,” Ph.D. dissertation, Dept. of Electrical and Computer Engineering, Monterey, CA, USA, 2018 [Online]. Available: http://hdl.handle.net/10945/67548

[86] K. P. Burnham and D. R. Anderson, *Model selection and multimodel inference : a practical information-theoretic approach*, 2nd ed. New York: Springer, 2002.

[87] M. R. Symonds and A. Moussalli, “A brief guide to model selection, multimodel inference and model averaging in behavioural ecology using akaike’s information criterion,” *Behavioral Ecology and Sociobiology*, vol. 65, no. 1, pp. 13–21, 2011.

[88] MathWorks, “Lognormal distribution - MATLAB & Simulink,” Accessed Apr. 26, 2021 [Online]. Available: https://www.mathworks.com/help/stats/lognormal-distribution.html

[89] MathWorks, “Beta distribution - MATLAB & Simulink,” Accessed Mar. 18, 2022 [Online]. Available: https://www.mathworks.com/help/stats/beta-distribution.html

[90] NIST, “1.3.6.6.17. Beta distribution,” Accessed Apr. 28, 2022 [Online]. Available: https://www.itl.nist.gov/div898/handbook/eda/section3/eda366h.htm

[91] A. Kim, “Conjugate prior explained,” Toward Data Science, Jan. 8, 2020 [Online]. Available: https://towardsdatascience.com/conjugate-prior-explained-75957dc80bfb

[92] MathWorks, “Gamma distribution - MATLAB & Simulink,” Accessed Mar. 18, 2022 [Online]. Available: https://www.mathworks.com/help/stats/gamma-distribution.html

[93] NIST, “1.3.6.6.11. Gamma distribution,” Accessed Apr. 28, 2022 [Online]. Available: https://www.itl.nist.gov/div898/handbook/eda/section3/eda366b.htm

[94] Britannica, “Gamma distribution,” Accessed Apr. 29, 2022 [Online]. Available: https://www.britannica.com/science/gamma-distribution

[95] J. Fridrich, *Steganography in digital media: principles, algorithms, and applications*. Cambridge ; New York: Cambridge University Press, 2010, oCLC: ocn422764994.

[96] Bitcoin, “Protect your privacy,” Accessed Jul. 19, 2019 [Online]. Available: https://bitcoin.org/en/protect-your-privacy

[97] OpenSig, “OpenSig: Digital signature technology,” Accessed Oct. 15, 2020 [Online]. Available: http://www.opensig.net/

[98] D. Jain, “LSB image steganography using Python,” The Startup, Aug. 22, 2021 [Online]. Available: https://medium.com/swlh/lsb-image-steganography-using-python-2bbbee2c69a2

[99] R. Cramer and V. Shoup, “Design and analysis of practical public-key encryption schemes secure against adaptive chosen ciphertext attack,” Cryptology ePrint Archive, Report 2001/108, 2001 [Online], https://ia.cr/2001/108.

[100] J. Katz, *Introduction to Modern Cryptography*, 3rd ed. (Chapman & Hall CRC cryptography and network security). Boca Raton, FL, USA: Chapman & Hall CRC, 2020.

[101] 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.

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

[103] 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.

[104] A. Beregszaszi, “EIP-2681: Limit account nonce to 2^64-1,” Ethereum Improvement Proposals, Apr. 25, 2020 [Online]. Available: https://eips.ethereum.org/EIPS/eip-2681

[105] G. Wood, “EIP-161: State trie clearing (invariant-preserving alternative),” Ethereum Improvement Proposals, Oct. 24 2016 [Online]. Available: https://eips.ethereum.org/EIPS/eip-161

[106] M. Harrigan and C. Fretter, “The unreasonable effectiveness of address clustering,” in *2016 Intl IEEE Conferences on Ubiquitous Intelligence Computing, Advanced and Trusted Computing, Scalable Computing and Communications, Cloud and Big Data Computing, Internet of People, and Smart World Congress (UIC/ATC/ScalCom/CBDCom/IoP/SmartWorld)*, 2016, pp. 368–373.

[107] Web3.py, “Introduction — Web3.py 5.28.0 documentation,” Accessed Jul. 19, 2019 [Online]. Available: https://web3py.readthedocs.io/en/stable/

[108] G. Ethereum, “Geth documentation | Go Ethereum,” Accessed Mar. 09, 2022 [Online]. Available: https://geth.ethereum.org/docs/

[109] PyCryptodome, “PyCryptodome 3.14.1 documentation,” Accessed Mar. 03, 2022 [Online]. Available: https://pycryptodome.readthedocs.io/en/latest/src/introduction.html

[110] G. Bertoni, M. Peeters, G. Van Assche *et al.*, “The keccak reference,” Keccak Team, Jan. 14, 2011 [Online]. Available: https://keccak.team/files/Keccak-reference-3.0.pdf

[111] G. Bertoni, J. Daemen, M. Peeters, and G. Van Assche, "Duplexing the sponge: Single-pass authenticated encryption and other applications," in *Selected Areas in Cryptography*, A. Miri and S. Vaudenay, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2012, pp. 320–337.

[112] 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*. NIST, 2010.

[113] N. Mouha, "NISTIR 8319. review of the advanced encryption standard," NIST, July 2021 [Online]. Available: https://nvlpubs.nist.gov/nistpubs/ir/2021/NIST.IR.8319.pdf

[114] J. Soto and L. Bassham, "NISTIR 6483. randomness testing of the advanced encryption standard finalist candidates," NIST, Apr. 2000 [Online]. Available: https://nvlpubs.nist.gov/nistpubs/Legacy/IR/nistir6483.pdf

[115] D. R. Brown, "Sec 2: Recommended elliptic curve domain parameters," *Standards for Efficient Cryptography*, Certicom Corp., Jan. 27, 2010 [Online]. Available: https://www.secg.org/sec2-v2.pdf

[116] M. Fersch, E. Kiltz, and B. Poettering, "On the provable security of (EC)DSA signatures," in *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security* (CCS '16). New York, NY, USA: Association for Computing Machinery, 2016 [Online], p. 1651–1662. Available: https://doi.org/10.1145/2976749.2978413

[117] V. Buterin, "EIP-2: Homestead hard-fork changes," Ethereum Improvement Proposals, Nov. 15, 2015 [Online]. Available: https://eips.ethereum.org/EIPS/eip-2

[118] Ethereum, "ethereum.crypto — Ethereum Specification documentation," Accessed May. 23, 2022 [Online]. Available: https://ethereum.github.io/execution-specs/autoapi/ethereum/crypto/elliptic_curve/index.html

[119] Bitcoin, "Transactions — Bitcoin," Accessed Nov. 29, 2021 [Online]. Available: https://developer.bitcoin.org/devguide/transactions.html

[120] MathWorks, "Fit probability distributions to data - MATLAB," Accessed Mar. 18, 2022 [Online]. Available: https://www.mathworks.com/help/stats/distributionfitter-app.html

[121] V. Pareto, "La legge della domanda," *Giornale degli Economisti*, vol. 10 (Anno 6), pp. 59–68, 1895 [Online]. Available: http://www.jstor.org/stable/23219874

[122] V. Pareto, *Cours d'économie politique*. Geneva, CHE: Librairie Droz, 1964, vol. 1.

[123] C. P. Bartels and H. Van Metelen, *Alternative probability density functions of income: A comparison of the lognormal-, Gamma-and Weibull-distribution with Dutch data*. Amsterdam, NLD: Vrije Universiteit, Economische Faculteit, 1975.

[124] A. B. Salem and T. D. Mount, "A convenient descriptive model of income distribution: the gamma density," *Econometrica: journal of the Econometric Society*, pp. 1115–1127, 1974.

[125] L. C. Thurow, "Analyzing the American income distribution," *The American Economic Review*, vol. 60, no. 2, pp. 261–269, 1970.

[126] J. Schwartz, R. T. Godwin, and D. E. Giles, "Improved maximum-likelihood estimation of the shape parameter in the Nakagami distribution," *Journal of Statistical Computation and Simulation*, vol. 83, no. 3, pp. 434–445, 2013.

[127] J. B. McDonald and M. Ransom, *The generalized Beta distribution as a model for the distribution of income: Estimation of related measures of inequality*. New York, NY: Springer New York, 2008 [Online], pp. 147–166. Available: https://doi.org/10.1007/978-0-387-72796-7_8

[128] Zoho, "Zoho Sign and blockchain - Electronic signature and digital signature software," Accessed Oct. 25, 2020 [Online]. Available: https://www.zoho.com/sign/features-and-benefits/blockchain.html

[129] Etherscan, "Ethereum daily transactions chart," Accessed on Mar. 13, 2021 [Online]. Available: https://etherscan.io/chart/tx

[130] MathWorks, "Unit root nonstationarity - MATLAB & Simulink," Accessed Feb. 04, 2021 [Online]. Available: https://www.mathworks.com/help/econ/unit-root-nonstationarity.html#bsf2u3f-14

[131] MathWorks, "Augmented Dickey-Fuller test - MATLAB adftest," Accessed Feb. 04, 2021 [Online]. Available: https://www.mathworks.com/help/econ/adftest.html#bta7rpp

[132] G. W. Schwert, "Tests for unit roots: A monte carlo investigation," *Journal of Business & Economic Statistics*, vol. 7, no. 2, pp. 147–159, 1989 [Online]. Available: https://www.tandfonline.com/doi/abs/10.1080/07350015.1989.10509723

[133] MathWorks, "KPSS test for stationarity - MATLAB kpsstest," Accessed Feb. 04, 2021 [Online]. Available: https://www.mathworks.com/help/econ/kpsstest.html

[134] D. Kwiatkowski, P. C. Phillips, P. Schmidt, and Y. Shin, “Testing the null hypothesis of stationarity against the alternative of a unit root: How sure are we that economic time series have a unit root?” *Journal of econometrics*, vol. 54, no. 1-3, pp. 159–178, 1992.

[135] J. H. McDonald, *Handbook of Biological Statistics*, 3rd ed. Baltimore, MD, USA: Sparky House Publishing, 2014.

[136] T. Pham and S. Lee, “Anomaly detection in the bitcoin system - a network perspective,” arXiv, 2016 [Online]. Available: https://arxiv.org/abs/1611.03942

[137] P. Jurkiewicz, G. Rzym, and P. Boryło, “Flow length and size distributions in campus internet traffic,” *Computer Communications*, vol. 167, p. 15–30, Feb 2021 [Online]. Available: http://dx.doi.org/10.1016/j.comcom.2020.12.016

[138] R. Agarwal, S. Barve, and S. K. Shukla, “Detecting malicious accounts in permissionless blockchains using temporal graph properties,” *Applied Network Science*, vol. 6, no. 1, p. 9, Dec. 2021 [Online]. Available: https://appliednetsci.springeropen.com/articles/10.1007/s41109-020-00338-3

[139] L. A. Adamic, R. M. Lukose, A. R. Puniyani, and B. A. Huberman, “Search in power-law networks,” *Physical Review E*, vol. 64, no. 4, Sep 2001 [Online]. Available: http://dx.doi.org/10.1103/PhysRevE.64.046135

[140] J. Poon and T. Dryja, “The bitcoin lightning network: Scalable off-chain instant payments,” Lightning Network, Jan. 14, 2016 [Online]. Available: https://lightning.network/lightning-network-paper.pdf

[141] V. K. Kanth, “Blockchain for use in collaborative intrusion detection systems,” M.S. thesis, Dept. of Electrical and Computer Engineering., NPS, Monterey, CA, USA, 2019 [Online]. Available: http://hdl.handle.net/10945/63465

[142] Google, “BigQuery API client libraries,” Accessed Apr. 04 2022 [Online]. Available: https://cloud.google.com/bigquery/docs/reference/libraries

[143] Google, “Getting started with authentication | Authentication,” Accessed Apr. 04 2022 [Online]. Available: https://cloud.google.com/docs/authentication/getting-started

THIS PAGE INTENTIONALLY LEFT BLANK

Initial Distribution List

1. Defense Technical Information Center
   Ft. Belvoir, Virginia

2. Dudley Knox Library
   Naval Postgraduate School
   Monterey, California