Lancaster University

School of Computing and Communications

Kleptography and Steganography in Blockchains

Nasser Alsalami

Under the Supervision of
Professor Bingsheng Zhang and
Doctor Antonios Gouglidis

Submitted for the degree of
Doctor of Philosophy
July 2020

Abstract

Despite its vast proliferation, the blockchain technology is still evolving, and witnesses continuous technical innovations to address its numerous unresolved issues. An example of these issues is the excessive electrical power consumed by some consensus protocols. Besides, although various media reports have highlighted the existence of objectionable content in blockchains, this topic has not received sufficient research. Hence, this work investigates the threat and deterrence of arbitrary-content insertion in public blockchains, which poses a legal, moral, and technical challenge. In particular, the overall aim of this work is to thoroughly study the risk of manipulating the implementation of randomized cryptographic primitives in public blockchains to mount kleptographic attacks, establish steganographic communication, and store arbitrary content.

As part of our study, we present three new kleptographic attacks on two of the most commonly used digital signatures: ring signature and ECDSA. We also demonstrate our kleptographic attacks on two real cryptocurrencies: Bytecoin and Monero. Moreover, we illustrate the plausibility of hijacking public blockchains to establish steganographic channels. Particularly, we design, implement, and evaluate the first blockchain-based broadcast communication tool on top of a real-world cryptocurrency. Furthermore, we explain the detrimental consequences of kleptography and steganography on the users and the future of the blockchain technology. Namely, we show that kleptography can be used to surreptitiously steal the users' secret signing keys, which are the most valuable and guarded secret in public blockchains. After losing their keys, users of cryptocurrencies will inevitably lose their funds. In addition, we clarify that steganography can be used to establish subliminal communication and secretly store arbitrary content in public blockchains, which turns them into cheap cyberlockers. Consequently, the participation in such blockchains, which are known to store unethical content, can be criminalized, hindering the future adoption of blockchains.

After discussing the adverse effects of kleptographic and steganographic attacks on blockchains, we survey all of the existing techniques that can defend against these attacks. Finally, due to the shortcomings of the available techniques, we propose four countermeasures that ensure kleptography and steganography-resistant public blockchains. Our countermeasures include two new cryptographic primitives and a generic steganography-resistant blockchain framework (SRBF). This framework presents a universal solution that deters steganography and practically achieves the right to be forgotten (RtbF) in blockchains, which represents a regulatory challenge for current immutable blockchains.

Declaration

I hereby declare that the work in this thesis has not been submitted for a degree at any other university. I also certify that the work is entirely my own, and where otherwise, it is duly acknowledged and clearly cited.

Signature:

Nasser Alsalami - October 26, 2020

Acknowledgement

This work would not have been completed without the help and support of many persons, to whom I would like to express my most profound appreciation. First and foremost, I would like to express my sincere and utmost gratitude to my supervisors, Prof. Bingsheng Zhang and Dr Antonios Gouglidis. Prof. Zhang's continuous guidance has greatly helped me in every step of this course. His constructive criticism and positive feedback have taught me beyond the subject of our research. Dr Gouglidis has contributed massively to the final year of my Ph.D. His generosity with his time and effort, which is beyond description, has always inspired me to do my best. I would also like to gratefully recognize the invaluable input and professional assistance of my postgraduate appraisal monitor Dr Alistair Brown. Besides, I wish to extend my gratitude to all of my colleagues in Security Lancaster and all of the staff in the School of Computing and Communications.

Words fall short for expressing my heartfelt thanks to my parents, whose love and support have motivated me throughout my studies and career. They are the role model that I strive to follow for their perfect selflessness and countless sacrifices. Moreover, I wish to thank my two kids, *Ahmed* and *Aseela*, whose innocence and playfulness have not only kept me going but made each day of this journey more joyful. They have had to stay away from their grandparents, cousins, and relatives, at times felt lonely, and had to put up with their continuously busy parents. For all of this, I sincerely thank them and hope that they can benefit from their experience during this time. Last but not least, I would like to wholeheartedly thank my wife, *Arwa*, who has made this work possible with her genuine altruism, unparalleled sincerity, and true love.

Publications

- [AZ20]: N. Alsalami and B. Zhang. Uncontrolled randomness in blockchains: Covert bulletin board for illicit activity. In *2020 IEEE/ACM International Symposium on Quality of Service (IWQoS)*, June 2020.
- [AZ19a]: N. Alsalami and B. Zhang. SoK: A systematic study of anonymity in cryptocurrencies. In *2019 IEEE Conference on Dependable and Secure Computing (DSC)*, pages 1–9, Nov 2019.
- [AZ19b]: N. Alsalami and B. Zhang. Utilizing public blockchains for censorship circumvention and IoT communication. In *2019 IEEE Conference on Dependable and Secure Computing (DSC)*, pages 1–7, Nov 2019.

Table of Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Overview

Since the inception of Bitcoin [Nak08] in 2009, the blockchain, Bitcoin's underlying technology, has proliferated various sectors [lUJ17], such as the energy sector [MNB+17], finance [Tap17], healthcare [AEVL16, Med20, MCAF20], logistics [KHD17], and the field of cryptocurrencies [Nak08, eth20, MGGR13]. Cryptocurrencies represent the first realization of the blockchain technology and remain to be the most popular blockchain applications.

The blockchain technology is a type of distributed ledger technologies (DLT), which eliminate the need for a trusted central authority by storing a digital ledger in a distributed manner. Any change in the data of a distributed ledger will result in simultaneously updating the ledger in all of the participating machines [DSLG17]. In contrast, blockchains are DLT's, which consist of an append-only ledger that is represented as *chained* blocks, which are sequentially and cryptographically linked using cryptographic hash functions. Blockchains are generally categorized according to their access permission to *permissioned* and *permissionless* blockchains, and according to their accessibility to *private* and *public* blockchains. Participation in public blockchains is available to anyone, whereas participation in private blockchains is governed by one or more parties. On the other

hand, permissionless blockchains permit any party that joins the blockchain to transact, create new blocks, and change the state of the ledger. On the contrary, permissioned blockchains restrict the parties that can transact and change the ledger [PCAP19].

Typically, the participating nodes in a blockchain application create signed transactions and broadcast them through their peer-to-peer (P2P) network. These transactions are later grouped in blocks that are appended to the chain by the *miners* who are nominated according to the used *consensus* protocol. The consensus protocol is essential to maintain a shared global view of the blockchain and ensure that the same ledger is replicated in all participating nodes. Examples of the different consensus protocols include the Proof-of-Work (PoW) protocol, where miners are randomly sampled in proportion to their computational power, and the Proof-of-Stake (PoS) protocol, where miners are nominated in proportion to their stake or wealth [Qua11]. In the following, for simplicity, we describe the process of generating transactions and blocks in the blockchain when using the PoW consensus protocol.

Blockchain users broadcast their data as *transactions* that propagate through the blockchain P2P network according to specific diffusion protocols [FV17a, FV17b]. The P2P nodes check the validity of the transactions and broadcast them to their neighbouring nodes. Eventually, the transactions are received by the *miners* who compete to solve a computational puzzle in the PoW consensus protocol. The miner, who wins the competition, groups several transactions into a *block* and appends it to the end of the ledger. A block is a collection of transactions, and each transaction contains one or more cryptographically signed inputs. For example, a Bitcoin transaction usually contains multiple inputs, and each input is signed using the ECDSA signature with the owner's private key(s) [NBF+16].

As shown in Table 1.1, most cryptocurrencies use randomized signature schemes to sign their transactions. Namely, 17 out of 20 cryptocurrencies that we surveyed use ECDSA and ring signature [RST01], which are randomized, i.e. non-deterministic, cryptographic signatures. This observation is central to this thesis. In particular, this thesis explores the risk of using randomized cryptographic primitives in public blockchains

				Anonymity Technique						
T	#	**crypto-currency**	**+CT** [1]	ECDSA EdDSA	**ring signature**	**mixers**	**commitments**	**ZKP** [2]	**bullet-proofs**	**stealth addressing**
Pseudo-anonymity	1	Bitcoin		✓						
	2	Ethereum		✓						
	3	Ethereum Classic		✓						
	4	Bitcoin Cash		✓						
	5	Bitcoin Diamond		✓						
	6	Litecoin		✓						
	7	Cardano		✓						
	8	IOTA		WT[5]						
	9	Dogecoin		✓						
	10	NEM		✓						
	11	Nano		✓						
	12	Lisk		✓						
	13	Waves		✓						
	14	Tether		✓						
	15	USD Coin		✓						
SA [3]	16	Dash			✓					
	17	Bytecoin			✓					✓
	18	Monero	✓		✓				✓	✓
FA [4]	19	Zerocoin				✓	✓	✓		
	20	Zcash	✓				✓	✓		

Table 1.1: Categorization of 20 cryptocurrencies and protocols according to their tier of privacy (T) and used techniques. (1) Confidential transactions. (2) Zero-knowledge proofs. (3) Set anonymity. (4) Full anonymity. (5) Winternitz signature. Note, although these 20 cryptocurrencies do not necessarily have the biggest market capitalization, they have been selected because of the accessible technical information about their transactions' signature schemes.

and their vulnerability to *kleptography* and *steganography*. These two types of attacks, as demonstrated in this thesis, might lead to mass-scale theft of cryptocurrency coins and facilitate the malicious dissemination and storage of arbitrary content. We focus on permissionless public blockchains for the following four reasons. Firstly, most, if not all, known cryptocurrencies, such as Bitcoin [Nak08] and Ethereum [eth20], are permissionless and public. Secondly, permissionless public blockchains represent a more lucrative target for distributing malicious content because they have a wider reach, compared to private blockchains. Thirdly, the organization(s) governing a private blockchain can prevent a known malicious node from joining the blockchain. Fourthly, the organization(s) governing a permissioned blockchain can restrict the ability of a dishonest node from altering the state of the ledger. For these reasons, we mainly consider permissionless public blockchains as they represent a more accessible and more fruitful target for saboteurs to launch kleptographic attacks and establish steganographic channels.

Gus Simmons, the pioneer of the field of steganography [Sim84], stated that 'the

realization that cryptography can be exploited to achieve malicious ends as easily as it can to achieve beneficial ones is a novel and valuable insight - to both designers and counter-designers of information security and integrity protocols' [YY04]. Although Simmon's finding may no longer be novel, it is still relevant today, especially in public blockchains, which are commonly open-source and cryptographically complex, in which we demonstrate how the very means of security can also be abused to endanger the security of the users and jeopardize the technology itself. Therefore, the main goals of this thesis are to study the threat of maliciously manipulating randomized cryptographic primitives in public blockchains to mount kleptographic attacks and implement steganographic channels, and design kleptography and steganography-resistant blockchains.

In the following, we describe the motivations behind this research, its aims, and objectives in Sec. 1.2. Then, we summarize the significant contributions of this work in Sec. 1.3. Finally, Sec. 1.4 illustrates the structure of this thesis.

1.2 Motivations, Aims, and Objectives

The threat of *kleptography* on randomized cryptographic primitives has received significant research [YY96, YY97a, BPR14, BJK15, AMV15]. The subject recently attracted more scrutiny due to some revelations of state-sponsored mass-surveillance [BBG13, BPR14, BJK15]. Kleptography studies the covert mis-implementation of secure cryptographic primitives to exclusively and subliminally steal the user's secrets while avoiding detection in the black-box setting [YY97b]. Nonetheless, despite the known and plausible threat of kleptography on cryptographic primitives, it has not been rigorously studied in public blockchains despite the following attributes that collectively make it more feasible in public blockchains as compared to other applications. First of all, the sheer cryptographic complexity in blockchains requires specialized expertise to scrutinize a given application for possible kleptographic backdoors. This complexity may obstruct the detection of unintentional flaws and intentional backdoors. Besides, it has been reported that the development of public blockchain applications is highly centralized with very few code contributors [AMM18], which facilitates bias. Furthermore, users tend to trust their

software and the underlying operating systems [Sch19]; they download and use pre-compiled executables, such as closed-source cryptocurrency wallets, without necessarily inspecting them for plausible mis-implementation. *Therefore, the first aim of this thesis is to explore the threat of kleptography on public blockchains.* To achieve this aim, we have the following research objectives. (1) Design new kleptographic attacks on commonly used randomized cryptographic primitives in public blockchains. (2) Implement these attacks in real-world blockchain applications. (3) Demonstrate how these attacks can lead to the secret exfiltration of the users' signing keys, which represent the users' most-guarded secret.

Furthermore, blockchains permanently store their content organized in *chained blocks* of data, and content can not be deleted except with newer copies of the ledger, i.e. hard forks [PDC17]. Although permanent storage is suitable for specific applications, such as cryptocurrencies, this feature infringes with the right-to-be-forgotten (RtbF) as required by law, e.g. the European General Data Protection Regulation [Eur16]. On the one hand, arbitrary-content insertion in the blockchain is the basis for certain benign applications, such as Tithonus [RC19] and Catena [TD17], which are censorship-circumvention and consensus agreement applications, respectively. Both Tithonus and Catena depend on inserting data in Bitcoin transactions. Also, R3C3 [MMSK18] is a censorship-bootstrapping tool that inserts non-financial data in Zcash [zca20]. On the other hand, the adverse effects of arbitrary content in blockchains outweigh its advantages. For example, malicious users may target blockchain platforms to store and disseminate objectionable content, as discussed by various media reports [S. 18, H. 18, BBC19, Sky18]. For instance, the BBC reported that egregious unethical images are stored in some cryptocurrencies, and highlighted the difficulties of identifying and removing such content from blockchain applications [BBC19]. Besides, Matzutt et al. reported that 0.8% of 146 million Bitcoin transactions store content on the blockchain or use non-standard scripts [MHH+16]. Later, Matzutt et al. surveyed the methods used to store non-financial content in Bitcoin [MHH+18]. They discovered that 1.4% of all Bitcoin transactions contain non-financial data, and retrieved over 1600 files, some of which contain immoral

content.

Intuitively, a user seeking to insert arbitrary content in blockchains can replace the redundant parts of a given transaction, such as the unused parts of the scripts in Bitcoin transactions [NBF+16]. However, this is detectable, as the redundant components have a known syntax while the user-inserted content is either human-readable if clear or random if encrypted. Also, specific to Bitcoin, a user can use certain Bitcoin transactions that are meant to be used for logging, non-financial communication, and storing content in the Bitcoin network [MHZ+18]. Nonetheless, this approach burns funds, offers limited bandwidth, and is economically infeasible. A user can alternatively insert arbitrary content using provably-secure subliminal techniques. This approach makes it hard to identify the embedded content, not to mention the infeasibility of removing it from the blockchain.

Moreover, it is widely known that randomized cryptographic primitives consume random input seeds or coins, and these coins can be replaced, in principle, by pseudo-random strings. This explains why randomized cryptographic primitives are considered to be the primary enabler of *kleptography* attacks [YY96, YY97a, BPR14, BJK15, AMV15]. Since these strings are embedded in a computationally-undetectable manner, they can also, intuitively, be used to establish *steganographic* communication [Sim84, And96, AP98]. Therefore, randomized cryptographic primitives that are used in blockchains present a plausible paradigm for storing arbitrary content. However, the extent of the threat that this paradigm represents has not received attention from the research community. *Consequently, this thesis's second aim is to study the threat of steganographic arbitrary-content insertion in public blockchains by maliciously manipulating their randomized cryptographic primitives.* We attempt to achieve this aim by focusing on the exploitation of randomized signature schemes in public blockchains. However, our findings are analogously applicable, beyond signatures, to similar randomized primitives. These include the commonly used non-interactive zero-knowledge proofs (NIZK). Note that randomized signature schemes present a lucrative target for content-insertion since they are extensively used in blockchains compared to other protocols such as IPsec or Transport

Layer Security (TLS), where signatures are mainly used in the authentication stage. Whereas, in blockchains, multiple signatures are usually used in every transaction. For example, a cryptocurrency transaction with several inputs typically contains a signature for each of the inputs [HAZ17].

As shall be demonstrated in this thesis, kleptography can be used to maliciously implement blockchain applications to surreptitiously leak the users' keys, which, in the context of cryptocurrencies, inevitably leads to mass-scale stealing of coins. Whereas, steganography can be used to secretly store arbitrary content in blockchains, and turn public blockchains into cheap cyberlockers. Since storing unethical content in blockchains can motivate regulators to criminalize the participation in such blockchains, steganographic channels represent a threat not only to the users but may also jeopardize the future adoption of the technology itself.

Given the aforementioned adverse effects of steganography and kleptography on blockchains' users and the future of the blockchain technology, it is vital to research new cryptographic schemes and blockchain designs that are resistant to kleptography and steganography. *Consequently, the third aim of this thesis is to investigate the current techniques that can resist steganography and design new steganography-resistant blockchains.* To meet this aim, we have the following two objectives. (1) Investigate and assess all of the available techniques that have been proposed in the literature to counter kleptography and steganography. (2) Overcome the shortcomings of the current techniques, if any, by proposing new efficient countermeasures.

1.3 Contribution

The contributions of this thesis can be summarized by the following four contributions.

1. **New kleptographic attacks on public blockchains.** In Chapter 3, we demonstrate the threat of *kleptography* in public blockchains. We show, for the first time, that kleptography can be used for mass-scale theft of users' keys, causing the users to lose their identity and cryptocurrency funds. In particular, we present three new kleptographic attacks on two of the most commonly-used signatures schemes in

cryptocurrencies: a kleptographic attack on the random numbers of the ring signature [RST01] and two kleptographic attacks on the ECDSA signature. These three kleptographic attacks can be used to subvert cryptocurrency wallets to steal the users' secret signing keys clandestinely. Notably, all of these attacks have the following properties:

(i) *Passive attacks.* After the victim user downloads and installs the subverted wallet, the attacker does not need to interact directly with the victim's wallet. The communication channel between the subverted wallets and the attacker is through the posted blockchain transactions.

(ii) *Undetectability.* The transactions generated by the subverted wallets are computationally indistinguishable from the honestly generated ones. Therefore, there is not any probabilistic polynomial-time (PPT) watchdog, which can detect these attacks in the black-box setting.

(iii) *Interoperability.* The subverted wallets transact seamlessly with normal wallets, i.e. they can send to and receive from other wallets regardless of whether other wallets are subverted.

(iv) *Subtlety.* Following the definition of *kleptography*, we consider our attack *exclusively* in the black-box setting. However, if optimized, the difference between a subverted wallet source code and the original code is only about ten lines of code in two functions. This subtlety makes it difficult for technology-savvy users to review and detect the kleptographic attack even if the subverted wallet is open-source.

Additionally, we implement and evaluate our attack on the ring signature in two real-world cryptocurrencies: Bytecoin (v 3.3.3) [Byt18] and Monero (v 0.12.0.0) [Mon18]. Besides, we implement and evaluate our attacks on ECDSA on an implementation of the signature rather than a blockchain application, because we lack funds for any cryptocurrency that uses ECDSA, such as Bitcoin, and other applications that use this signature are not open-source.

2. **The first blockchain-based broadcast communication tool.** In Chapter 4, we

demonstrate the plausibility of hijacking public blockchains to communicate steganographically through the exploitation of randomized signature schemes. Specifically, we design, implement, and evaluate the security and performance of the first-ever blockchain-based broadcast-communication application that we refer to as *Skywhisper*. With its provable security, high throughput per transaction, and low transaction fees, *Skywhisper* also facilitates storing arbitrary content in public blockchains and turns them into hidden cheap and uncensorable cyberlockers. At the time of submission, persistently storing 1GB of data on Skywhisper costs less than \$11.5[1].

3. **Surveying and assessing all of the existing techniques to defend against kleptography and steganography in public blockchains.** We present in Chapter 5 all of the previously proposed cryptographic techniques and new blockchain designs, and assess their effectiveness as countermeasures against kleptography and steganography. We specifically assess the existing techniques according to three metrics: their efficiency as protection against kleptographic stealing of secret information, their ability to prevent steganographic communication, and their ability to resist the persistent storage of arbitrary content.
4. **Proposing efficient countermeasures.** Due to the shortcomings of the existing techniques, we propose in Chapter 6 four new countermeasures, including two new cryptographic primitives and a generic steganography-resistant blockchain framework. The newly proposed countermeasures ensure the resistance of public blockchains to kleptographic and steganographic abuse. Our countermeasures can be used as the basis for future steganography-resistant blockchain designs.

Besides, since our kleptographic and steganographic attacks mainly exploit primitives that are used to achieve anonymity, we surveyed in [AZ19a] the various anonymity guarantees in cryptocurrencies. Notably, we presented a novel three-tier categorization of anonymity in cryptocurrencies and public blockchains in general. Our categorization approach is based on two factors. (1) The ability of the used anonymity scheme to break any possible linkage between transactions. (2) The ability of the used scheme

[1] The cost is based on the price of Bytecoin (BCN) as shown on `https://coinmarketcap.com/` on 26/June/2020 .

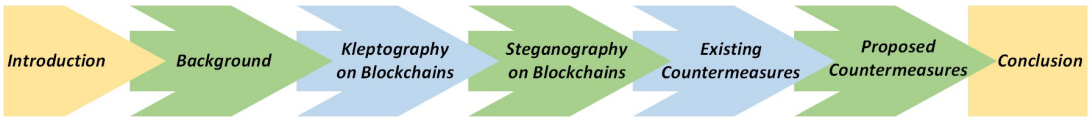


Figure 1.1: Thesis roadmap

to hide users' identities. Considering these two factors, we defined the following three tiers of anonymity in cryptocurrencies: (1) *pseudonymity*, (2) *set anonymity*, and (3) *full anonymity*. Also, there is a very notable feature that is used by some cryptocurrencies in conjunction with these three levels of anonymity, which we refer to as *confidential transactions*. This feature ensures the transacted amounts are hidden. Table 1.1 shows 20 cryptocurrencies categorized according to the aforementioned tiers of anonymity. This table also clarifies the techniques that each cryptocurrency employs to meet its anonymity guarantees. These techniques are mainly the following six techniques: (1) ECDSA/EdDSA signature, (2) ring signatures, (3) mixers, (4) commitments, (5) zero-knowledge proofs, and (6) stealth addressing. Importantly, 17 cryptocurrencies of the 20 surveyed currencies use randomized signature schemes. A detailed description of our survey on the privacy in cryptocurrencies [AZ19a] is available in Appendix A.

1.4 Thesis Structure

Fig. 1.1 illustrates the structure of this thesis. Chapter 2 defines the notations used in this thesis, introduces some preliminary concepts, and discusses the related work. Since randomized cryptographic primitives are susceptible to subversion attacks in the form of *kleptography* [YY97a] and *steganography* [Sim84], and since most of the anonymity techniques in public blockchains are randomized, as demonstrated in Table 1.1 [AZ19a], Chapter 3 introduces three kleptographic attacks on two of the most widely used digital signatures in cryptocurrencies. Additionally, Chapter 3 demonstrates the realization of our attacks in two real cryptocurrencies: Monero [Mon18] and Bytecoin [Byt18]. These attacks illustrate the extent of the threat that kleptography poses to users' secret information and cryptocurrency funds.

After that, Chapter 4 explains the adverse effects of blockchain-based steganographic

communication on the users and the technology, and illustrates how public blockchains can be hijacked to establish subliminal communication. In particular, we present a new broadcast communication application on top of the real-world cryptocurrency of Bytecoin [Byt18]. The proposed application, called *Skywhisper*, is provably secure, economically efficient, and has a broadcast channel of up to 64 subscribers.

After demonstrating kleptography and steganography on public blockchains and discussing their detrimental effects, Chapter 5 presents all of the current techniques that can be used as deterrence against kleptography and steganography. Besides, we assess the effectiveness of the surveyed techniques as a defence against the attack scenarios in Chapters 3 and 4. Due to shortcomings in the available techniques, Chapter 6 proposes four efficient countermeasures, including a generic framework for future steganography-resistant blockchains. Finally, Chapter 7 concludes this thesis and discusses directions for future work.

Chapter 2

Background

This chapter describes background information that is relevant to understanding the work of this thesis. Sec. 2.1 explains the used notations throughout this document. After that, Sec. 2.2 discusses some preliminary concepts, such as the notion of cryptocurrencies and cryptocurrency wallets. Finally, Sec. 2.3 discusses the related work to this thesis: *kleptography*, *steganography*, and *content-insertion* in blockchains.

2.1 Notations

In the following, we describe some of the notations that are relevant to understanding the content of this thesis. If s is a string, then we use $|s|$ to denote its length, and use $s_{[a,b]}$ to denote a subset of this string starting from the a^{th} bit to the b^{th} bit. Also, if $\mathcal{S}$ is a set, then $|\mathcal{S}|$ represents the size of the set, i.e. the number of elements in $\mathcal{S}$. We use $\mathbb{Z}_p$ and $\mathbb{Z}_p^*$ to denote a group of prime order p and a group of prime order p without 0, respectively. When an element s is randomly picked from a set $\mathcal{S}$ we use $s \xleftarrow{\$} \mathcal{S}$. We also use $\{0,1\}^\ell$ to represent a set of binary strings of length ℓ, and $\{0,1\}^*$ to represent the set of all binary strings of arbitrary length.

We denote by $\mathsf{poly}(\cdot)$ a polynomially-bounded function and by $\mathsf{negl}(\cdot)$ a negligible function. If A is used to denote an algorithm, then $b \leftarrow \mathsf{A}(a)$ represents that A outputs b when given a as input. If A is a randomized algorithm, then $b \leftarrow \mathsf{A}(a; r)$ denotes that

A outputs b when given a and randomness r as input. We say that an algorithm A runs in probabilistic polynomial-time (PPT) if it is a randomized algorithm and for any input $a, r \in \{0,1\}^*$, the execution of $\mathsf{A}(a;r)$ terminates in at most $\mathsf{poly}(|x|)$ steps. If an algorithm A is a PPT algorithm, e.g. an attack, then A is also said to be *efficient*. When A represents a keyed algorithm, such as encryption and decryption algorithms, then $\mathsf{A}_z(x)$ denotes that A operates on the string x under the key z. For example, $\mathsf{ENC}_z(x)$ denotes an encryption algorithm ENC that encrypts x under the key z. Similarly, we use $\mathsf{A}(1^\lambda)$ to denote that A takes as input a security parameter of length λ.

2.2 Preliminary Concepts

In the following subsections, we provide an overview description of the preliminary concepts that are relevant to the content of this thesis.

2.2.1 Cryptocurrencies

Various forms of digital cash had been proposed before the current cryptocurrencies. For example, Chaum described blind signatures for untraceable payments in 1983 [Cha83] and later proposed untraceable electronic cash in which a central entity, i.e. a bank, actively participates to issue cash, ensure the validity of transactions, and prevent double-spending [CFN90, MLS+15]. However, cryptocurrencies were initiated more recently in 2009 after the inception of Bitcoin [Nak08], which has become the basis for numerous other cryptocurrencies, such as Ethereum[1] and Litecoin[2]. In the following, we describe cryptocurrencies by the example of Bitcoin.

Unlike Chaum's cash, Bitcoin is a decentralized cryptocurrency where nodes in the P2P network create their transactions, broadcast, validate, group them into blocks, and mine new blocks according to the consensus protocol without the need for any third party. Nodes create their public-private key pairs and transact with each other using their pseudo-anonymous *addresses* that are associated with their public keys. These

[1] `https://ethereum.org`
[2] `https://litecoin.org`

public keys are published in the Bitcoin network, allowing the public verification of all transactions that are signed using the corresponding private keys [Nak08].

An essential aspect of Bitcoin's design is the so-called 'Nakamoto Consensus'. Namely, Bitcoin solves the Byzantine generals problem [LSP82] and prevents double-spending through the 'Nakamoto Consensus' [lUJ17, Nak08], which relies on the Proof-of-Work protocol and the 'longest-chain win' rule. The latter rule means that nodes choose the longest ledger, i.e. choose the ledger with more mined blocks, when more than one ledger exists. In the proof-of-work protocol, the miners compete against each other to solve a computationally-intensive puzzle by randomly choosing a nonce number so that the cryptographic hash value is less than or equal to a predefined value. The miner who succeeds in solving the puzzle is rewarded with a mining reward, which represents how new coins are minted in Bitcoin [NBF+16]. Since Bitcoin is fully decentralized and miners are sampled in proportion to their computational power, Bitcoin relies on the assumption that honest participants correspond to at least 51% of the total computational power in the P2P Bitcoin network [MLS+15].

2.2.2 Cryptocurrency wallets

A cryptocurrency wallet is an abstraction of the object where the cryptocurrency's credentials reside. Usually, these credentials are in the form of the user's private key(s) and their corresponding public keys. Cryptocurrency wallets are generally categorized into the following categories according to how the keys are managed [Gur18]. (1) *Software wallets* are applications that store the public-private keys and manage the user's transactions. If the keys are stored at the client-side, then the wallet is said to be a *client-side* wallet. When the keys are stored at a trusted third party and accessed online, it is called a *web-based* wallet. Software wallets can also be stored in air-gapped machines to further protect the keys such as Ellipal cold wallet [Ell20]. (2) *Hardware wallets* store the keys in dedicated and trusted Swiss-Army-like off-line hardware modules, which are usually PIN-protected and provide USB connectivity. Examples of such wallets include Tezor [Sat20] and Ledger wallets [Led20]. (3) *Paper wallets* store the keys on a printed

paper, usually in the form of alphabetical strings or QR code. Paper wallets are similar to *brain wallets* in which users are required to memorize some mnemonic corresponding to their keys [Gur18].

2.2.3 Pseudo-random functions (PRF)

Let us define a function F that takes as input a key $k \in \mathcal{K}$ and a string $x \in \mathcal{X}$, and outputs $y \in \mathcal{Y}$. $\mathcal{K}$ is the key space, $\mathcal{X}$ is the input space, and $\mathcal{Y}$ is the output space, i.e. $F : \mathcal{K} \times \mathcal{X} \rightarrow \mathcal{Y}$ or simply $y = F_k(x)$. Also, let us denote by $\mathcal{F}_{\{\mathcal{X},\mathcal{Y}\}}$ the set of *all* functions f's that map inputs from $\mathcal{X}$ to $\mathcal{Y}$. F is said to be a pseudo-random function (PRF) if its output is computationally indistinguishable from the output of a randomly-sampled function $f \xleftarrow{\$} \mathcal{F}_{\{\mathcal{X},\mathcal{Y}\}}$ [LR86, AMV15]. More specifically, assume an oracle $\mathcal{O}$ flips a coin to randomly choose $b \xleftarrow{\$} \{0, 1\}$, then:

- if $b = 0$, $\mathcal{O}$ randomly chooses a key $k \xleftarrow{\$} \mathcal{K}$, and executes $r \leftarrow F_k(x)$, and
- if $b = 1$, $\mathcal{O}$ randomly chooses a function $f \xleftarrow{\$} \mathcal{F}_{\{\mathcal{X},\mathcal{Y}\}}$, and executes $r \leftarrow f(x)$.

In addition, assume that a PPT adversary $\mathcal{A}$ is given access to the oracle $\mathcal{O}$ and is challenged to guess b. In particular, $\mathcal{A}$ chooses $x \in \mathcal{X}$, and queries $\mathcal{O}$. $\mathcal{A}$ is then given r and asked to generate a bit $\acute{b}$ representing his guess. F is considered to be a secure PRF if there does not exist any PPT adversary that can distinguish between the two outputs except for a negligible probability ϵ, i.e. the following is valid: $\Pr[b = \acute{b}] - 1/2 \leq \epsilon$.

2.2.4 Collision-resistant hash functions (CRH)

A hash function H is a function that maps a message of arbitrary length $x \in \mathcal{X}$ to an output of fixed length that is called the hash value or the message digest $y \in \mathcal{Y}$, i.e. $H : \mathcal{K} \times \mathcal{X} \rightarrow \mathcal{Y}$, where $\mathcal{K}$ is the key space. A collision happens if two different messages m_1 and m_2 hash to the same value, that is, $H(m_1) = H(m_2)$, such that $m_1 \neq m_2$. Usually, the size of the domain $|\mathcal{X}|$ is larger than the size of the range $|\mathcal{Y}|$; hence, collisions are unavoidable. However, H is said to be collision-resistant if it is computationally hard to find a collision. In other words, if H is a collision-resistant hash (CRH) function, then a PPT adversary $\mathcal{A}$ can not find two distinct messages that hash to the same value except

for a negligible probability ϵ, i.e. $\Pr[(m_1, m_2) \leftarrow \mathcal{A} \wedge m_1 \neq m_2 \wedge H(m_1) = H(m_2)] \leq \epsilon$.

Secure hash functions should also satisfy the following two properties. *Pre-image resistance*, also called one-wayness, implies that given a hash value h, it should be computationally infeasible to find a message m, such that $H(m) = h$. Whereas the other property, *second pre-image resistance* means that given a message m, there does not exist any computationally-bounded adversary that can find a different message $\acute{m}$ that hashes to the same value as m.

2.2.5 Encryption algorithms

An (asymmetric) encryption scheme consists of three algorithms $(\mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec})$ as follows [ADR02]:

- $(\mathrm{PK}, \mathrm{SK}) \leftarrow \mathsf{KeyGen}(1^\lambda)$ is a key generation algorithm that takes as input a security parameter 1^λ and outputs a pair of public and secret keys $(\mathrm{PK}, \mathrm{SK})$.
- $\mathsf{ct} \leftarrow \mathsf{Enc}_{\mathrm{PK}}(m)$ is a *probabilistic* encryption algorithm that takes as input a public key PK, and a message m from the message space $\mathcal{M}$, and outputs a ciphertext ct.
- $m \leftarrow \mathsf{Dec}_{\mathrm{SK}}(\mathsf{ct})$ is a decryption algorithm that takes as input a secret key SK, the ciphertext ct, and outputs the corresponding plaintext message m.

It is required that, except for a negligible probability, $m \leftarrow \mathsf{Dec}_{\mathrm{SK}}(\mathsf{Enc}_{\mathrm{PK}}(m))$. In this work, we frequently refer to the notion of semantic security of encryption schemes, or indistinguishability of ciphertext under chosen-plaintext attack (IND-CPA). This notion means that there does not exist any PPT adversary $\mathcal{A}$, who is given a randomly-generated public key PK and can distinguish with a non-negligible probability between the encryption of two messages m_0 and m_1 that are chosen by $\mathcal{A}$. In other words, a given encryption scheme is IND-CPA secure if it satisfies the following statement, where $\mathsf{negl}(\lambda)$ is a negligible function in the security parameter λ [ADR02].

$$\Pr\left[b = \acute{b} \;\middle|\; \begin{array}{c} (\mathrm{PK}, \mathrm{SK}) \leftarrow \mathsf{KeyGen}(1^\lambda), \{m_0, m_1\} \in \mathcal{M} \leftarrow \mathcal{A}(\mathrm{PK}), \\ b \xleftarrow{\$} \{0,1\}, \mathsf{ct}_b \leftarrow \mathsf{Enc}_{\mathrm{PK}}(m_b), \acute{b} \leftarrow \mathcal{A}_{\mathrm{PK}}(\mathsf{ct}_b) \end{array}\right] - \frac{1}{2} \leq \mathsf{negl}(\lambda)$$

2.2.6 Digital signatures

In the following, we use Pointcheval and Stern's definition of generic signature schemes [PS96]. A user with a public-secret key pair $(\mathrm{PK}, \mathrm{SK})$ can sign a message m with his secret key SK to generate a signature σ. The signature σ can be verified publicly by any party using the signer's public key PK. However, it is hard for any other party to forge a user's signature on m without his SK, which is kept secret by the user. More formally, a signature scheme $\mathcal{S}$ is a tuple of three algorithms $(\mathsf{KeyGen}, \mathsf{Sign}, \mathsf{Verify})$ as follows:

- $(\mathrm{PK}, \mathrm{SK}) \leftarrow \mathsf{KeyGen}(1^\lambda)$ is a key generation algorithm that takes as input a security parameter 1^λ and outputs a pair of public and secret keys $(\mathrm{PK}, \mathrm{SK})$.
- $\sigma \leftarrow \mathsf{Sign}(\mathrm{SK}, m)$ is a signing algorithm that takes as input a secret key SK, and a message m, and outputs a signature σ.
- $b \leftarrow \mathsf{Verify}(\mathrm{PK}, m, \sigma)$ is a verification algorithm that takes as input the signer's public key PK, the signed message m and the signature σ, and outputs $b = 1$ if σ is valid.

It is required that for any m in the message space: $\mathsf{Verify}(\mathrm{PK}, m, \mathsf{Sign}(\mathrm{SK}, m)) \rightarrow 1$. Also, a signature scheme is considered *secure* if it achieves existential unforgeability under chosen-message attack (EUF-CMA) [GMR88, ADR02, BB04]. EUF-CMA means that there does not exist any PPT adversary $\mathcal{A}$, who can forge a valid signature except for a negligible probability. $\mathcal{A}$'s probability in forging a valid signature $\sigma_\mathcal{A}$ is defined according to the following security experiment. $\mathcal{A}$ is given a public key PK and access to a signing oracle to sign q messages $\{m_1, \ldots, m_q\}$. After the query phase, $\mathcal{A}$ is challenged to generate a signature $\sigma_\mathcal{A}$ for a message $m_\mathcal{A}$ that has not been queried to the signing oracle, i.e. $m_\mathcal{A} \notin \{m_1, \ldots, m_q\}$. $\mathcal{A}$ wins in this experiment if $\sigma_\mathcal{A}$ is valid, i.e. $1 \leftarrow \mathsf{Verify}(\mathrm{PK}, m_\mathcal{A}, \sigma_\mathcal{A})$ [ADR02]. In the following two subsections, we describe two signature schemes that are particularly relevant to the work in this thesis: the ECDSA signature, and the *ring signature*.

ECDSA (KeyGen, Sign, Verify)

KeyGen(param):

- Pick random $s \xleftarrow{\$} \mathbb{Z}_q$;
- Set $S = g^s = (S_x, S_y)$;
- Output $(\mathrm{PK} = S, \mathrm{SK} = s)$;

Sign(s, m):

- Pick random $r \xleftarrow{\$} \mathbb{Z}_q$;
- Set $R = g^r = (R_x, R_y)$;
- Set $w = (\mathsf{hash}_p(m) + s \cdot R_x) \cdot r^{-1} \pmod q$;
- Output $\sigma = (R_x, w)$;

Verify(S, m, σ):

- $u_1 = \mathsf{hash}_p(m) \cdot w^{-1} \pmod q$ and $u_2 = R_x \cdot w^{-1} \pmod q$;
- Compute $P = g^{u_1} S^{u_2} = (P_x, P_y)$;
- Output valid if and only if $P_x \stackrel{?}{=} R_x \pmod q$;

Figure 2.1: ECDSA signature scheme: g is a generator of the curve, and hash_p is a secure collision-resistant cryptographic hash function

2.2.6.1 ECDSA

The ECDSA signature is a randomized-signature scheme over the NIST elliptic curves that has been widely used in blockchains, particularly among pseudo-anonymous currencies (Table 1.1), such as Bitcoin, Ethereum, etc. ECDSA is a tuple of three algorithms (KeyGen, Sign, Verify), and is based on the elliptic curve discrete-logarithm problem. An illustration of the ECDSA signature is depicted in Fig. 2.1.

2.2.6.2 Ring signature

Ring signatures were introduced by Rivest et al. [RST01], extending the idea of *group signatures* that was proposed by Chaum and van Heyst [CH91]. In a group signature, there is a trusted group manager who constructs the signature and can de-anonymize the other members. On the contrary, ring signatures do not have managers, and any signer can sign on behalf of the group. A ring signature scheme consists of a tuple of four algorithms $\mathcal{S} = (\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Sign}, \mathsf{Verify})$ as follows:

- $\mathsf{param} \leftarrow \mathsf{Setup}(1^\lambda)$ is a setup algorithm that takes as input a security parameter 1^λ and outputs a system parameter param.

- $(\mathrm{PK}, \mathrm{SK}) \leftarrow \mathsf{KeyGen}(\mathsf{param})$ is a key generation algorithm that takes as input the setup parameter param, and outputs a pair of public and secret keys $(\mathrm{PK}, \mathrm{SK})$.
- $\sigma \leftarrow \mathsf{Sign}(\mathcal{P}, \mathrm{SK}, \ell, m)$ is a signing algorithm that takes as input a set of public keys $\mathcal{P} = \{\mathrm{PK}_1, \ldots, \mathrm{PK}_k\}$, which corresponds to the k ring members, i.e. the possible signers, the secret key SK of the actual signer, an index ℓ such that SK is the secret key of the ℓ^{th} member, and a message m, and outputs a signature σ. The ring signature is said to be of size k.
- $b \leftarrow \mathsf{Verify}(\mathcal{P}, m, \sigma)$ is a verification algorithm that takes as input a set of public keys $\mathcal{P}$, the signed message m and the signature σ, and outputs $b = 1$ if and only if the signature is valid.

Besides the general soundness and completeness properties of digital signatures, ring signature guarantees signer-ambiguity, where a verifier should not be able to identify the actual signer with probability greater than $1/k$. Besides, as explained by Rivest et al. [RST01], ring signatures are 'set-up free' in the sense that the signer does not need the consent nor the participation of the other members to include them in the ring signature. The signer only needs to know the members' public keys. Since their introduction, ring signatures have evolved in three directions: (1) *threshold* ring signatures [BSS02], (2) *linkable* ring signatures [LWW04, ACST06, LW05], and (3) *traceable* ring signatures [FS07, Fuj11]. As ring signatures are particularly pertinent to the work in this thesis, we clarify below the usage of ring signatures in the CryptoNote protocol and in Monero's RingCT.

The use of ring signatures in CryptoNote. CryptoNote[3] is an open-source framework that is implemented by various cryptocurrencies, such as Bytecoin [Byt18]. As shown in its whitepaper [Sab13], CryptoNote's signature uses a slightly modified version of the traceable ring signature scheme proposed by Fujisaki et al. [FS07]. According to this protocol, the payer generates a one-time public key $R = g^r$ and computes the address $T = (g^{\mathsf{hash}_p(A^r)} \cdot B)$. In this case, the payee can compute the corresponding one-time private key as $t = \mathsf{hash}_p(R^a) + b$. Note that the one-time ring signature scheme

[3] `https://cryptonote.org`

CryptoNote Ring Signature

$\mathsf{Sign}(\{\mathrm{PK}_j\}_{j=1}^k, \mathrm{SK}_\ell, \ell, m)$:

- Set $I = \mathsf{hash}_g(\mathrm{PK}_\ell)$;
- For $j \in [k]$, pick $q_j \xleftarrow{\$} \mathbb{Z}_p$;
- For $j \in [k], j \neq \ell$, pick $w_j \xleftarrow{\$} \mathbb{Z}_p$;
- For $j \in [k]$:
  - Set $L_j = g^{q_j}$ if $j = \ell$;
    Set $L_j = g^{q_j} \cdot \mathrm{PK}_j^{w_j}$ if $j \neq \ell$;
  - Set $R_j = (\mathsf{hash}_g(\mathrm{PK}_j))^{q_j}$ if $j = \ell$;
    Set $R_j = (\mathsf{hash}_g(\mathrm{PK}_j))^{q_j} \cdot I^{w_j}$ if $j \neq \ell$;
- Set $c = \mathsf{hash}_p(m, L_1, \ldots, L_k, R_1, \ldots, R_k)$;
- For $j \in [k]$:
  - Set $c_j = w_j$ if $j \neq \ell$;
    Set $c_j = c - \sum_{j=1}^k c_j$ if $j = \ell$;
  - Set $r_j = q_j$ if $j \neq \ell$;
    Set $r_j = q_\ell - c_\ell \mathrm{SK}_\ell$ if $j = \ell$;
- Return $\sigma = (I, c_1, \ldots, c_k, r_1, \ldots, r_k)$.

$\mathsf{Verify}(\{\mathrm{PK}_j\}_{j=1}^k, m, \sigma)$:

- For $j \in [k]$:
  - Set $L'_j = g^{r_j} \cdot \mathrm{PK}_j^{c_j}$;
  - Set $R'_j = (\mathsf{hash}_g(\mathrm{PK}_j))^{r_j} \cdot I^{c_j}$;
- Check if $\sum_{j=1}^k c_j \stackrel{?}{=} \mathsf{hash}_p(m, L'_1, \ldots, L'_k, R'_1, \ldots, R'_k)$

Figure 2.2: CryptoNote ring signature: signing and verification algorithms

is transformed from the OR-composition of Schnorr's identification Sigma protocols. Also, the protocol has an LNK algorithm to link any two signatures produced by the same signing key, which is essential to prevent double-spending. In particular, let $\mathrm{PK}_\ell = g^{\mathrm{SK}}$ be the signer's public key, and define $I = (\mathsf{hash}_g(\mathrm{PK}_\ell))^{\mathrm{SK}}$ as a 'key image' as part of the signature. The ring signatures signed by the same secret key would have an identical key image; therefore, double-spending can be defeated efficiently by simply checking if the key image has already been used. For clarity, CryptoNote's ring signature is shown in Fig. 2.2. The signing algorithm takes as input $\mathcal{P} = \{\mathrm{PK}_j\}_{j=1}^k$ which is a set of public keys, the signer's secret key SK_ℓ, index $\ell \in [k]$, and the message m to be signed. As seen in Fig. 2.2, the verifier does not know any information beyond the fact that 1 out of the possible k signers generated the signature σ.

Borromean ring signature in Monero's RingCT. Monero[4] is a popular cryptocurrency that was initially based on CryptoNote, and Borromean ring signature is a *1-out-of-n* signature invented by Maxwell and Polestra [MP15] that is an optimization of the AOS ring signature by Abe et al. [AOS02]. Borromean ring signature is used in Monero's ring confidential transactions (RingCT) to generate *rangeproofs* by creating a ring signature for each digit of the committed amount. This effectively hides the committed amount a while proving its range $a \in [0, 2^{31} - 1]$. If an amount a is encoded in 16 base-4 digits $d_0 d_1 d_2 \dots d_{15}$, the sender chooses 16 blinding factors x_i and generates 16 commitments, one for each digit as follows:

$$C_i = x_i G + a_i H$$

where $a_i = (4^{15-i} * d_i)$, $i \in [0, 15]$ and $d_i \in [0, 3]$. After that, the sender generates 4 public keys $C_{i,d}$ for each digit d_i corresponding to the 4 possible values $d \in \{0, 3\}$ as follows:

$$C_{i,d} = C_i - (4^{15-i} * d)H$$

This will generate four public keys $C_{i,d}$ for each ring signature of which the signer/sender knows *one* private key, x_i corresponding to the public key that was generated for the actual committed value of the digit. For example, if the 3^{rd} most significant digit of the base-4-encoded a has a value of 1, that is $d_2 = 1$, then the signer would know the private key x_2 corresponding to the second public key in the 3^{rd} ring: $C_{2,1}$ because:

$$C_{2,1} = C_2 - (4^{15-2} * 1)H$$

$$C_{2,1} = (x_2 G + (4^{15-2} * 1)H) - (4^{15-2} * 1)H = x_2 G$$

By choosing the blinding factors $x_0, x_1, \dots, x_{15}$ so that they add up to x, which is the blinding factor used for the overall commitment C, any party can publicly verify that $C = C_0 + C_1 + \dots + C_{15}$. However, no one can know which of the possible four values each commitment corresponds to, nor can they know which value in the range is committed to.

[4] `https://monero.org`

2.3 Related Work

The work in this thesis is closely related to the topics of *kleptography*, *steganography*, and *arbitrary-content insertion in blockchains*. The relevant literature review of each of these three topics is presented respectively in Sec. 2.3.1, Sec. 2.3.2, and Sec. 2.3.3.

2.3.1 Kleptography

In 1996, Young and Yung introduced the notion of kleptography [YY96, YY97a], which has also been referred to as *Algorithm-Substitution Attacks (ASA)* [BPR14, BJK15], and *Subversion Attacks* (SA) [AMV15].

Besides pioneering the topic of kleptography, Young and Yung laid the foundations for the study of the malicious implementation of cryptography in a series of publications [YY96, YY97a, YY97b, YY98, YY03, YY05b, YY05a, YY05c, YY06, YY07, YY10, YY04]. In [YY96], the authors shed light on the risk of using cryptographic schemes as black boxes and presented the notion of Secretly-Embedded Trapdoor with Universal Protection (SETUP). *A SETUP attack is when the manufacturer or developer of these schemes maliciously implements them to exclusively get the user's secret in an undetectable fashion in the black-box setting, without the need for explicit subliminal channels.* They described a set of SETUP attacks on specific primitives, such as the RSA key generation algorithm, El-Gamal, and DSA. They also presented some recommendations to mitigate the effects of SETUP attacks like controlling the randomness, cascading independently-developed cryptographic systems, and checking the integrity of cryptographic software. In [YY97a], Young and Yung referred to this new area of threats as *kleptography* and introduced new definitions for their SETUP attacks. They defined a *weak* SETUP as an attack that is not detectable except for the attacker and the owner of the subverted hardware or software who owns his private key. In contrast, a *strong* SETUP is when the attack is only detectable to the attacker even if the users who own the secret key reverse-engineer the subverted system. They also showed a new kleptographic attack on the Diffie-Hellman key exchange protocol based on the discrete-logarithm problem. Besides, they strengthened their previous attack on RSA by using a technique called

'probabilistic bias removal' to ensure the uniform distribution and the indistinguishability of the cryptographic elements that are kleptographically-leaking the RSA private key to the attacker. After that, Young and Yung showed broader applicability of their SETUP attacks and the general vulnerability of discrete-log problem (DLP)-based cryptosystems to kleptography. In particular, they described a generic kleptographic attack on the discrete-log problem, and then applied it to multiple DLP-based cryptosystems, such as the ElGamal encryption and signature algorithm, DSA, and the Schnorr signature algorithm [YY97b].

Unlike their previous work, which focused on designing kleptographic attacks on public-key cryptosystems, Young and Yung attempted to design kleptographic attacks on *secret* symmetric ciphers in [YY98, YY03, YY05c]. In [YY98], the authors presented a new deterministic and symmetric block cipher called 'Monkey', whose specifications are secret, i.e. not publicly available, to highlight the risk of trusting black-box block ciphers. To learn the user's symmetric key k, Monkey requires that the attacker, i.e. the malicious designer, obtains a sufficient number of ciphertexts under the same key k, each containing one known plaintext bit. Monkey was improved later in [YY03], which describes a new secret symmetric cipher called Black Rugose that, unlike Monkey, eliminates the need for known plaintext and leaks more than one bit of the user's key in each ciphertext c. The number of ciphertexts needed to leak an n-bit key in Black Rugose is $((n \log n)/b)$, where b is the number of leaked bits in each ciphertext. Note, both Monkey and Black Rugose are secret ciphers and are in direct violation of Kerckhoffs' principle, which states that the security of a cryptosystem should be based on the difficulty of determining the secret key and not from the secrecy of the algorithm itself [PP10]. Besides, Young and Yung presented the first general-purpose subliminal channel that is built into a secret symmetric cipher in [YY05c]. Their subliminal channel is unique in two regards. Firstly, it is built onto a deterministic function, unlike previous subliminal channels, which were built onto probabilistic cryptosystems. Secondly, the subliminal channel described in their work is atypical as both the sender and the receiver know beforehand the subliminal message m_s, and the sender subliminally and randomly transmits a bit b in each cipher

to the receiver who knows some private key. This uniqueness can limit the practicality of the presented subliminal channel.

Young and Yung also revisited their attack on RSA key generation [YY96, YY97a] in [YY05a, YY06, YY10]. In [YY05a], the authors focused on improving the undetectability of their attack on RSA key generation that was first presented in [YY96, YY97a]. Motivated by the fact that their previous attack on RSA [YY96] is no longer secure since the embedded attacker's public key is 512 bits, and RSA-576 has been factored [Wei03], Young and Yung proposed in [YY06] a new attack on RSA-1024 key-generation algorithm using a pair of twisted elliptic curves. Their attack is premised on employing elliptic curve Diffie-Hellman (ECDH) key exchange between the subverted RSA device and the attacker to allow him to factor the RSA modulus, and using point compression for space efficiency. Their key-recovery mechanism can also be used to enable legitimate escrow authorities to recover users' private keys. In [YY10], Young and Yung further improved on their previous work on RSA [YY06, YY07] by presenting a new space-efficient information-hiding algorithm called 'covert key exchange', which they used to minimize the information needed to be leaked and reduce the error probability. They used this algorithm to implement the first asymmetric kleptographic backdoor in the standard model, as opposed to the oracle mode, in RSA key generation. They also pointed out other possible applications of this algorithm, such as designing a kleptographic backdoor in SSL.

In light of the recent revelations regarding the use of algorithm-substitution attacks (ASA) [BBG13], i.e. kleptography, for mass surveillance, Bellare, Paterson, and Rogaway (BPR) [BPR14] formally studied the resistance of symmetric encryption algorithms against such attacks. They formalized the two security goals of the attacker who mounts ASA attacks: undetectability and surveillance resistance. The former indicates the ability of passive observers to distinguish ciphertext that is produced by the corrupted algorithm from that produced by the original algorithm. In contrast, surveillance resistance reflects the ability of the 'big brother', i.e. the attacker, who possesses the kleptographic secret key, in making this distinction. They presented two types of attacks: *IV-replacement* attacks

that apply to algorithms that surface the IV, and *biased-ciphertext* attacks, which are more generic and can be mounted on any stateless randomized cipher with enough entropy. BPR concluded that randomized stateless schemes are vulnerable to kleptography; whereas, deterministic stateful algorithms, such as unique-ciphertext encryption algorithms, are resistant against kleptography. Later, Bellare, Jaeger, and Kane (BJK) [BJK15] extended the work of BPR to attack stateful algorithms and introduced the notion of strong undetectability. Also, Degabriele et al. [DFP15] refined the security definitions presented in BPR and relaxed the requirement for perfect decryptability. Moreover, Bellare and Hoang [BH15] presented secure public-key encryption schemes that are resistant to the subversion of random number generators.

This thesis is particularly related to the topic of kleptography in signature schemes, which we describe in the following. The work of Ateniese et al. [AMV15] showed that randomized signature schemes are inevitably susceptible to subversion attacks, i.e. kleptography. Besides, they showed that schemes with unique signatures are kleptography-resilient. The authors also suggested the use of trusted cryptographic reverse firewalls [MSD15] to sanitize re-randomizable signature schemes.

Goh et al. [GBPG03] added an undetectable key-recovery mechanism to the implementation of SSL/TLS [DA99] and SSH2. The premise of their attack is that a malicious implementer, or escrow agency according to their terminology, can replace the random-looking text in these protocols by encryption of the session key. Since the output of semantically-secure encryption algorithms is indistinguishable from the pseudo-random-looking text, the output of the subverted protocols cannot be distinguished from the output of standard protocols except for the malicious implementer who possesses the recovery key. Another attack on SSL/TLS is presented by Gołebiewski et al. [GKZ06].

Also, a set of symmetric backdoored designs for RSA were presented by Crépeau and Slakmon [CS03] to secretly establish a subliminal channel in the public modulus (n, e) so that the attacker can factor n to compute the private modulus d. The authors employ the Coppersmith's algorithm [Cop96] to minimize the amount of the information to be leaked and speed up the process.

Examples of other kleptographic attacks include attacks on pseudo-random generators [DGG+15], DL-based signatures [Teş19], and DSA [BSKC19]. Teşeleanu [Teş19] described a threshold kleptographic attack on the generalized ElGamal signature that can be extended to similar DL-based signatures. They also listed some of the techniques proposed in the literature to defend against kleptography. Dodis et al. [DGG+15] formally studied how to subvert the Dual EC pseudo-random generators (PRG) and backdoored PRGs, generally. Besides, they studied how to immunize potentially subverted PRGs. Also, Hartl et al. [HAZ17] showed the existence of a subliminal channel in the EdDSA signature [BDL+11], explained its applicability in three scenarios, and discussed three possible mitigation and detection techniques. However, they concluded that none of their countermeasures are viable in network protocols. Besides, the work of Schnier et al. [SFKR15] provides a categorization of cryptographic techniques that are used secretly to weaken cryptography.

Rijmen and Preneel [RP97] presented constructions for backdoored block ciphers with hidden structures, called trapdoors. These backdoored ciphers look secure to any party, who does not know about the trapdoors, while allowing the attacker(s), i.e. the malicious designers, to obtain information about the key by using a small number of plaintexts. Their work helps demonstrate the possibility of such structures even in known algorithms, and the need to justify the design of pseudo-random generators. Later, the security of these trapdoored ciphers was broken in [WBDY98].

To thwart kleptography, several techniques have been proposed [YY96, BPR14, AMV15]. Young and Yung proposed the use of trusted external randomness sources, cascading independent cryptosystems, and checking the integrity of the cryptographic software [YY96]. Whereas, Bellare et al. argued that randomized schemes are inevitably susceptible to kleptography, while deterministic cryptographic schemes are resistant to such threat [BPR14, BJK15]. Furthermore, the integrity of the process of randomness generation can be checked using verifiable random functions [MRV99] and the controlled randomness technique proposed by Hanzlik et al. [HKK17].

In the context of subversion-resistant signatures, Zhang et al. [ZLLZ13] proposed

a subliminal-free variant of the Schnorr signature [Sch91] using an honest-but-curious *interactive warden*. Likewise, Bohli et al. [BGVS07] proposed a subliminal-free variant of ECDSA that requires *non-interactive wardens*. Besides, Ateniese et al. [AMV15] proposed the use of *reverse firewalls* to re-randomize the output of possibly sabotaged signature schemes. In addition, Russell et al. [RTYZ16a] modelled and proved that a full domain hash-based signature scheme achieves subversion resilience. Recently, Russell et al. [RTYZ16b, RTYZ17] proposed the use of a splitting-randomness technique to secure a randomizable IND-CPA public-key encryption. Furthermore, Fischlin and Mazaheri [FM18] proposed a novel technique that proactively defends against kleptographic attacks assuming initial *temporary trust*, i.e. subversion happens after a period of an honest initial phase. Using this initial phase of trust, they provided kleptography-resistant constructions for homomorphic public-key encryption, symmetric-key encryption, signature schemes, and physically unclonable function PUF-based key exchange. For more details, Chapter 5 discusses all of the existing techniques that have been proposed as countermeasures against kleptography in cryptographic schemes.

2.3.2 Steganography

The concept of *steganography* was introduced by Simmons' *prisoner's problem* [Sim84]. Simmons discussed the problem where two prisoners want to exchange secret information, an escape plan, without being detected by the prison's warden. The warden carefully inspects the exchanged messages and will throw any suspicious communication. In this context, the problem of *steganography* is the ability of the two prisoners to communicate secret information by embedding it in normal warden-inspected messages, known as *cover text*, without being caught by the warden. Cover texts that contain steganographically-hidden information are known as *stegotexts*.

Anderson et al. [And96, AP98] formally defined steganography and discussed the difficulty associated with formalizing a general proof of security for steganography in practice. Several works [Cac98, OME98, Mit00] provided information-theoretic treatment of steganography security and robustness. Cachin [Cac98] used information theory and

hypothesis testing to model the security of a stegosystem against a passive attacker as the relative entropy between the distribution of innocent cover text and that of stegotexts. According to this model, a steganographic system is said to be perfectly secure against passive adversaries if this relative entropy is zero, and ϵ-secure if the value is less than or equal to ϵ. Similarly, Ettinger [Ett98] used game theory to model active attackers on steganographic systems.

Unlike prior information-theoretic work that defined the security of steganographic systems based on hypothesis testing [Cac98], Zöllner et al. [ZFK+98] used information theory to model the security of stegosystems by the relative entropy of the hidden message, the entropy of the stegotext, and the entropy of the cover text. In particular, they argued that a stegosystem is information-theoretically secure if there is not any party that can learn any information about the hidden message by scrutinizing the stegotext and cover text. Zöllner et al. concluded the following two requirements for secure stegosystems: (1) the stegosystem key should be kept secret, and (2) the cover text should be unknown by sampling random cover texts.

Furthermore, Mittelholzer [Mit00] attempted to present a general model to assess the security and robustness of steganographic systems using information theory and based on mutual information. Namely, according to Mittleholzer's model, the security of a given steganographic system is assessed by the mutual information between the secret hidden message and the stegotext in which it is contained. Whereas, the robustness of a steganographic system is assessed by the mutual information between the secret message and the modified stegotext.

More recently, Hopper et al. provided a cryptographic formalization of steganographic security and robustness [HLvA02, HvL09]. The authors argue that information theory is limited when addressing security in steganography just as it is limited when addressing security in cryptography. Hence, Hopper et al. [HLvA02, HvL09] used cryptography and complexity-theory techniques to define symmetric-key steganographic systems that are secure against a passive adversary in terms of the *computational* indistinguishability of stegotext from the cover text. The authors also defined robustness in steganography as

the ability to resist the changes that are introduced by the adversary, i.e. the warden in this case, to the stegotext.

Katzenbeisser and Petitcolas presented a different approach to define security in the steganographic system [KP02]. Motivated by the work of Moskowitz et al. [MLC01], the authors argued that prior information-theoretic definitions, [ZFK+98, Cac98, Mit00], may not be appropriate to model the security of practical steganographic systems. Therefore, they defined the security of a given steganographic system $\mathcal{ST}$ based on a probabilistic game between the attacker and a judge. Namely, according to their model, $\mathcal{ST}$ is considered secure if the attacker, after a query phase, can not distinguish between a stegotext and an innocent cover text better than random. The authors also introduced the notion of 'conditional security' in steganographic systems.

The work of Petitcolas et al. [PAK99] gives an overview of the topic of steganography, explains relevant terminologies, and describes some of the used techniques and their respective known attacks. In addition, steganography has been applied in various media. Early steganography transmitted information by embedding it directly into the text, e.g. in music scores [PAK99]. Also, images [MBR99] and audio/video [Gop03, CXT06] can be used as cover texts to communicate steganographically. However, the work in this thesis is mainly related to steganographic attacks on cryptographic protocols where information is hidden in subliminal channels in these protocols [HAZ17].

2.3.3 Content insertion in blockchains

Metzutt et al. [MHH+16] provided insight regarding the various ways that could be exploited to store, possibly illegal, content in the Bitcoin blockchain. They listed four methods for embedding content in Bitcoin transactions: (1) including up to 100 bytes of arbitrary content in *coinbase* and OP_RETURN transactions, which offer an intended mechanism to augment Bitcoin transactions with arbitrary text, (2) replacing the public key (hash) in pay-to-pubkey (pay-to-pubkey-hash), (3) attaching up to 83 bytes in *nulldata* transactions, and (4) using non-standard scripts by, e.g. adding non-effective lines to the script. Furthermore, using some heuristics to analyse the plaintext of 146

million transactions, the authors of [MHH+16] reported that 0.8% transactions store non-financial content in the blockchain or use non-standard scripts. Later, Matzutt et al. [MHH+18] attempted to analyse the non-financial content in Bitcoin's blockchain. They surveyed the methods and services that are used to store non-financial content and provided a general categorization of the objectionable content that could be found in Bitcoin's blockchain. They discovered that 1.4% of all Bitcoin transactions contain non-financial data, and retrieved over 1600 files, some of which contain objectionable content. Nonetheless, some benign applications rely on content insertion in Bitcoin. For example, Tithonus [RC19] offers a Bitcoin-based censorship-resistant system, and Catena [TD17] is an application that uses Bitcoin OP_RETURN transactions to establish consensus among users on an application-specific log. The more recent work of Minaei et al. [MMSK18] presented a Zcash-based censorship-bootstrapping tool and explored content insertion techniques in Bitcoin, Zcash, Monero, and Etheruem.[5]

The work of Frkat et al. [FAZ18] showed how to insert arbitrary content in Bitcoin's transactions by replacing the ephemeral randomness in each transaction's ECDSA signature. The authors demonstrated their technique in the context of botnets, where a central bot, or botmaster, communicates commands subliminally to other bots in the botnet. Their model suffers a severe security vulnerability represented in the inability of the botmaster to generate the same command twice. Otherwise, the botmaster's private key can be computed by any observer. Besides, a watchdog can detect this steganographic communication when the botmaster communicates the same message more than once, i.e. their scheme is not secure against chosen plaintext-attack. Namely, a warden can detect the steganographic communication by repeating a message twice.

Moreover, Sawrd et al. surveyed some Bitcoin content-insertion techniques [SVS18]. They focused on replacing different parts of the various transaction scripts, such as replacing the public key with arbitrary data in the pay-to-public-key (P2PK) script and replacing the hash string in the pay-to-public-key-hash (P2PKH) script.

[5] [MMSK18] describes a technique to embed content steganographically in Monero's transactions' signatures that is similar to our technique. However, we note that an earlier version of this part of our work, which was eventually published in IWQoS'20, was submitted to USENIX'18 on 8^{th} Feb. 2018 before the publication of [MMSK18].

While there is a relatively significant body of research on *content insertion* in Bitcoin's blockchain [HvL09, MHH$^+$16, MHZ$^+$18, Shi14], Partala [Par18] was the first to discuss the use of *steganography* to *covertly* communicate in Bitcoin's blockchain. However, due to its limitation, Partala [Par18] considers their attack to be a proof of concept rather than a practical attack, as shall be seen in detail in Chapter 4.

Several techniques have been proposed to achieve long-term sustainability and minimize storage in blockchains; however, some of these techniques are also useful as deterrence against arbitrary content storage, and can effectively thwart malicious content. For example, various emerging blockchain frameworks store balances rather than the full transactions, e.g. PascalCoin [MS17] and Cryptonite [Min18]. Also, redactable blockchains [AMVA17] can be used to rewrite and remove malicious content from the blockchain.

Moreover, Matzutt et al. [MHZ$^+$18] proposed three complementing countermeasures to fight arbitrary content insertion in blockchains. Their first countermeasure is to use *content filters* to detect and reject unwanted content, e.g. rejecting a transaction if its 20-byte destination address has 18 printable characters. The second approach is to increase the transaction fee, which renders content insertion economically infeasible for large transactions. The third approach is to substitute addresses in Bitcoin's transactions with *self-verifying* address commitments. For example, instead of sending an address a, c_a is sent in the transaction, where $c_a = (G^a, r, \mathsf{Sign}(G^a||r, a))$, $r = \mathsf{CRC32}(t_1||\ldots||t_i)$, and t_i is the transaction corresponding to the i^{th} input.

Chapter 3

Kleptography in Blockchains

This chapter introduces the concept of *kleptography* and explains its significance in blockchains in Sec. 3.1, and Sec. 3.2, respectively. Then in Sec. 3.3 and Sec. 3.4, we describe three kleptographic attacks on two of the most widely used signature schemes in blockchains: CryptoNote's ring signature, and ECDSA. After the description of our kleptographic attacks, we explain the adverse scenarios of these attacks. In particular, Sec. 3.5 illustrates how these kleptographic attacks can secretly steal the users' confidential information. Finally, Sec. 3.6 demonstrates the realization of our attacks in two real cryptocurrencies, Bytecoin and Monero, and on an implementation of ECDSA.

3.1 Kleptography

Young and Yung first introduced the notion of kleptography in 1996 [YY96, YY97a]. Kleptography is defined as the malicious act of secretly replacing a secure algorithm with a sabotaged algorithm that surreptitiously leaks the user's confidential information exclusively to the attacker while avoiding detection in the black-box setting. Since its introduction, kleptography has also been referred to as *Algorithm-Substitution Attacks (ASA)* [BPR14, BJK15] and *subversion attacks* (SA) [AMV15]. Subsequent work demonstrated the possible use of kleptography in mass surveillance, and the susceptibility of all randomized symmetric encryption schemes to such attacks [BJK15, BH15]. Another demonstration of kleptographic attacks is found in the work of Goh et al. [GBPG03],

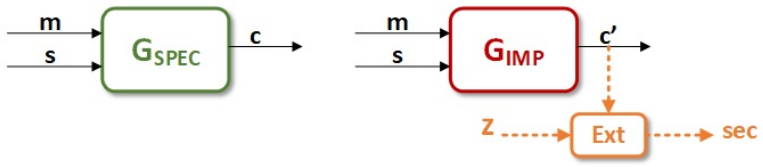


Figure 3.1: Kleptography: specification $\mathsf{G}_{\mathsf{SPEC}}$ takes as input a message m and a secret s, and outputs c. The malicious implementation $\mathsf{G}_{\mathsf{IMP}}$ takes the same inputs as the specification algorithm $\mathsf{G}_{\mathsf{SPEC}}$; however, it outputs a subverted ciphertext $\acute{c}$, which can leak some secret sec exclusively to the attacker who knows z. The leaked secret sec can be the user's secret key s or any other confidential information.

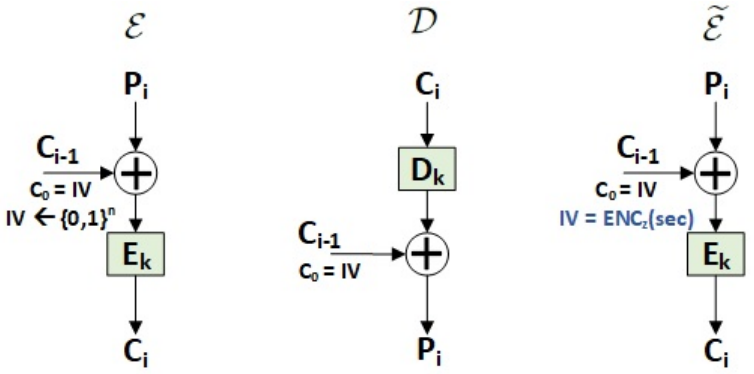


Figure 3.2: IV replacement [BPR14] is an example of kleptographic attacks. The randomly-generated initial vector IV of a CBC-mode encryption algorithm $\mathcal{E}$ is replaced in the subverted implementation $\widetilde{\mathcal{E}}$ by an encipherment of the victim's secret information; $\mathsf{IV} = \mathsf{ENC}_z(k)$. The adversary can extract k by intercepting IV and decrypting it; $k = \mathsf{DEC}_z(\mathsf{IV})$.

which presented practical hidden key-recovery attacks against the SSL/TLS and SSH2 protocols by modifying the implementation of the OpenSSL library. For consistency, we henceforth use the term *kleptography* instead of subversion attacks and ASA.

Informally, in kleptogrpahy, an adversary maliciously tampers with the implementation of a cryptographic algorithm $\mathsf{G}_{\mathsf{IMP}}$ and changes it from its specification $\mathsf{G}_{\mathsf{SPEC}}$ algorithm. The kleptographic implementation $\mathsf{G}_{\mathsf{IMP}}$ aims to: (i) subliminally and exclusively leak the user's secret information to the adversary, and meanwhile (ii) evade detection in the black-box setting by producing subverted output $\acute{c}$ that is computationally indistinguishable from normal output c. As in Fig. 3.1, the kleptographic implementation $\mathsf{G}_{\mathsf{IMP}}$ of the algorithm $\mathsf{G}_{\mathsf{SPEC}}$ allows the adversary, given their secret key z, to detect the subverted ciphertext $\acute{c}$ and extract the user's secret sec. Kleptographic attacks are significant due to their undetectability in the black-box setting and their detrimental

effects on the confidentiality of the users.

A basic example of kleptographic attacks is the *IV-replacement attack* proposed by Bellare et al. [BPR14] in the context of mass surveillance. In a simplified form, the adversary in this attack replaces a symmetric encryption system $\mathcal{SE} = (\mathcal{K}, \mathcal{E}, \mathcal{D})$ by a subverted implementation $\widetilde{\mathcal{SE}} = (\mathcal{K}, \widetilde{\mathcal{E}}, \mathcal{D})$, where $\mathcal{K}$ is a key-generation algorithm, $\mathcal{E}$ is an encryption algorithm, and $\mathcal{D}$ is a decryption algorithm. Assuming that $\mathcal{SE}$ operates in the Cipher Block Chaining (CBC) mode, as shown in Fig. 3.2, then the attacker can maliciously implement $\mathcal{E}$ so that it replaces the random initial vector IV by encryption of the user's secret sec under the attacker's secret key z; $\mathsf{IV} = \mathsf{ENC}_z(\mathsf{sec})$. Hence, the attacker can intercept the ciphertext and decrypt IV with their secret key z to reveal the user's secret sec, i.e. $\mathsf{sec} = \mathsf{DEC}_z(\mathsf{IV})$. If the leaked secret is the user's symmetric encryption key k, then by obtaining k, the attacker can decrypt all of the ciphertexts. Bellare et al. have also shown that the subverted IV in this example attack is computationally indistinguishable from randomly-generated IV, given that the attacker's encryption algorithm ENC is a secure pseudo-random function PRF [BPR14].

Kleptographic attacks on signature schemes. As the work in this chapter is particularly related to kleptographic attacks on signature schemes, in the following, we explain kleptography in the context of signatures. Young and Yung [YY97b] showed that DSA signature schemes could be subverted to leak secret information. Another kleptographic attack was proposed by the work of Teşeleanu [Teş19], which describes a threshold kleptographic attack on the generalized ElGamal signature that can be extended to similar discrete log-based signatures.

Since randomized cryptographic primitives are susceptible to kleptography as concluded by Bellare et al. [BPR14], theoretically, all algorithms ($\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Sign}, \mathsf{Verify}$) in a signature scheme $\mathcal{S}$, except Verify, which is usually deterministic; can be subverted to leak secret information. However, in practice, most blockchain platforms do not generate their setup parameters themselves; instead, widely trusted setup parameters, such as in the ED25519 curve, are adopted. Therefore, we do not consider kleptographic attacks on the Setup algorithm. Also, KeyGen algorithms are usually based on a one-way function,

Kleptography security experiment

$\mathbf{Expt}^{\mathsf{SU}}_{\mathcal{W}}(1^\lambda)$

1. $\mathcal{C}$ picks $b \xleftarrow{\$} \{0,1\}$;
2. $\mathcal{C}$ sends $(\mathrm{PK}, \mathrm{SK})$ to $\mathcal{W}$;
3. **For** $j \in \{1, \ldots, k\}$, $\mathcal{W}$ queries $\mathcal{C}$ for signature on $m_j \in \mathcal{M}$ and obtains
   - $\sigma_j \leftarrow \mathsf{Sign}(\mathrm{SK}, m_j)$ if $b = 0$;
   - $\sigma_j^* \leftarrow \mathsf{Sign}^*(\mathrm{SK}, m_j)$ if $b = 1$;
4. $\mathcal{W}$ outputs a bit b';
6. $\mathcal{C}$ returns $b \stackrel{?}{=} b'$;

Figure 3.3: Kleptography security experiment on subverted signatures between a challenger $\mathcal{C}$ and a watchdog $\mathcal{W}$, who wins in this security game when $b = b'$. The kleptographically-subverted algorithm Sign^* is said to be *secretly undetectable* if given $(\mathrm{PK}, \mathrm{SK})$, the probability that $\mathcal{W}$ distinguishes between genuine signatures and subverted signatures is negligible. If $\mathcal{W}$ is only given the public key (PK) in step 2 of this experiment, then the subverted scheme is said to be *publicly undetectable*.

and it is possible to leak $O(\log \lambda)$ bits through rejection sampling. Nevertheless, for most signature schemes, this will not provide sufficient bandwidth[1]. Therefore, this work focuses on the kleptographic attacks on the Sign algorithms of randomized signature schemes. As a result, we use the following definition of undetectability that is based on the definitions presented by Ateniese et al. [AMV15].

Public and secret undetectability. A kleptographic attack is said to be undetectable if there does not exist any PPT watchdog $\mathcal{W}$ that can win the security experiment in Fig. 3.3 with a non-negligible probability. In other words, a kleptographic attack is undetectable if there is not any PPT watchdog $\mathcal{W}$, who can distinguish whether a signature is produced by a subverted signing algorithm or the genuine one, except for a negligible probability.

Definition 3.1. *Let $\mathcal{S} = (\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Sign}, \mathsf{Verify})$ be a signature scheme. Let $\mathcal{M}$ be the message space. We say a subverted Sign^* algorithm is secretly undetectable if for any PPT watchdog $\mathcal{W}$, all $\{\mathrm{PK}, \mathrm{SK}\}$ output by $\mathsf{KeyGen}(\mathsf{param})$, and any integer $\lambda \in \mathbb{N}$, the advantage of $\mathcal{W}$ in winning the security experiment in Fig. 3.3 is negligible, i.e.:*

$$\mathsf{Adv}^{\mathsf{SU}}_{\mathcal{W}}(1^\lambda) = \left| \Pr\left[\mathbf{Expt}^{\mathsf{SU}}_{\mathcal{W}}(1^\lambda) = 1\right] - \frac{1}{2} \right| = \mathsf{negl}(\lambda)$$

[1] See *leakage resilient signatures* in [KV09, BSW11] for more discussion.

We say a subverted Sign^* *algorithm is publicly undetectable if* $\mathcal{W}$ *only receives* $\{\mathrm{PK}\}$ *in step 1 of Fig. 3.3.*

3.2 Significance of Kleptography in Blockchains

Before delving into the details of our kleptographic attacks on blockchains, it is essential to note that kleptographic attacks merit scientific research and should be carefully considered when designing and implementing cryptographic schemes, and that 'dismissing kleptography as far-fetched is naïve' [BPR14]. Likewise, despite the common open-source nature of public blockchains, kleptographic attacks are practically plausible and have severe repercussions on blockchains. Their plausibility and significance in blockchains are better seen in the light of the following attributes.

Firstly, cryptocurrencies, and blockchains in general, have very complex cryptographic primitives and mathematical structures, which may lead to unseen kleptographic attacks. In particular, this sheer complexity means very few experts are competent enough to assess their implementation [BL17], which may result in undetected flaws. As noted by Young and Yung [YY05a], despite the intuitive assumption that kleptographic attacks may only apply to black-box cryptography, it is uncommon for code, even when made available, to be sufficiently inspected. As an example of their argument, Young and Yung stated that in Eurocrypt 2004 a major implementation bug was revealed in an open-source signature scheme, in which obtaining a single signature is enough to reveal the secret signing key [Ngu04]. Moreover, the infamous Debian's flawed pseudo-random number generator [SFKR15] is an example of the, probably honest, flaws in the implementation of cryptographic primitives. This flaw was not detected for two years despite its open-source code. Another example of such flaws is shown by Knockel et al. [KRC18], who stated that the Tencent's QQ browser was using the so-called schoolbook RSA algorithm with no padding, which is well-known to be semantically insecure as it is a deterministic encryption scheme [PP10]. This is further demonstrated by Luu et al. [LCO+16], who noted that over 1/3 of the open-source smart contracts contain at least one bug, and some of them are maliciously embedded and can be

triggered later by the attackers in a similar manner to the widely-known Ethereum DAO hack [Sie16]. In addition, while experimenting with Bytecoin [Byt18], which is an open-source cryptocurrency, we noticed that the transactions' signatures diverge from their specifications in a way that can severely cripple the users' anonymity. This discovered bug is explained in more detail in Appendix B.

Secondly, although blockchain applications employ standard and well-established cryptographic schemes, adversaries may circumvent the security by resorting to secretly weakening these schemes through misimplementation. This makes kleptography a viable alternative for attackers. This is evident in the widely-publicized Snowden's revelations, which shed light on the fact that even resourceful organizations, such as the NSA, may not be able to break the security of standardized schemes except through secret trapdoors [BL17]. Adversaries may also resort to manipulating the standardization processes, as is the case with the Dual EC_DRBG [SFKR15]. Hence, the existence of such trapdoors should be investigated and thwarted by devising new kleptographic-resistant cryptographic primitives.

Thirdly, many cryptocurrencies are marketed as decentralized projects, yet studies have found that the development of many blockchain applications is highly centralized. Although there is not any evidence of malpractice, nor is the intention to make unsubstantiated accusations, this high centralization may cause bias and introduce intentional and unintentional flaws. An example of this highly-centralized development is that 30% of the source files in Bitcoin are written by a single author, and 7% of the code is contributed by the same author [AMM18]. Similarly, 20% of the source code in Ethereum is attributed to the same author [AMM18].

The fourth reason is that most end-users lack the ability and the means to check the conformity of executable applications, such as cryptocurrency wallets, with their reference source code. Hence, detecting implementation discrepancies is practically impossible. Besides, there is not any known methodology for a closed-source implementation to prove to the end-users that it follows specifications. As argued by Goh et al. [GBPG03], who implemented a hidden key-recovery mechanism

in the SSL/TLS protocol, it is not feasible for a black-box implementation to prove to the outside world that it honestly follows the original protocol and does not implement their key-recovery mechanism. In fact, in some platforms, such as iOS devices, users can not directly access the binary files without jailbreaking their devices, which paradoxically is not advisable and may render the device unsafe to run a cryptocurrency wallet. Also, it is uncommon for users to compile the source code of any application by themselves; instead, they usually rely on downloading readily-compiled executable applications. The difficulty of examining the implementation of a cryptocurrency wallet is even more pertinent to hardware wallets, such as the various Swiss-army-knife-like hardware wallets [Giz17]. It is practically impossible to audit the integrity of their implementation through the standard functionality 'correctness' test by observing input/output pairs in a black-box manner. Nonetheless, users are expected to blindly trust these wallets.

The fifth reason is the ease of distributing a kleptographically-sabotaged blockchain application, making the attack practically feasible. For example, an attacker can implement a kleptographic attack in an open-source wallet and market it as a closed-source wallet under a new name. The attacker can make his subverted wallet more appealing to unsuspecting users by making it user-friendly. Alternatively, an attacker can infect the victim users with some malware that replaces the user's wallet with the kleptographically-modified wallet. Such malware infection is plausible as Kaspersky Lab reported that there were over one million Bitcoin-wallet-stealing malware infections every month in 2013 [Lab14].

The sixth reason is attributed to the broadcast nature of the P2P blockchain network; hence, an attacker does not need to spoof the network actively. The attacker passively receives all transactions as all other nodes. On the contrary, Goh et al.'s hidden key-recovery in SSL/TLS requires the attacker to continuously spoof the network traffic [GBPG03]. Therefore, kleptographic attacks are evidently more attainable in the context of blockchain applications. **Likewise, the seventh reason is the ability of the attacker to stay anonymous due to the robust anonymity techniques in blockchains, as demonstrated in Table 1.1**.

Anonymity guarantees may encourage more attackers to mount kleptographic attacks to ensure avoiding detection and thus evading any legal prosecution.

Furthermore, the eighth reason for the significance of kleptographic attacks stems from their severe ramifications on the users of blockchains. The least repercussion could be the de-anonymization of the victim user(s). However, users can also lose their most-guarded secret in blockchains, their signing keys. In the context of cryptocurrencies, losing the secret keys will inevitably lead to the theft of the users' crypto funds, i.e. coins.

Due to the plausibility above and the potential threat of kleptographic attacks on blockchains, this chapter describes three new kleptographic attacks on public blockchains. In particular, we present attacks on two of the most commonly-used randomized signature schemes in blockchains: the ring signature and the ECDSA signature scheme.

3.3 Kleptographic Attack on Ring Signature

Many cryptocurrencies use ring signatures to protect users' privacy. For example, the CryptoNote framework [Sab13], which is adopted by around 20 cryptocurrencies[2], uses ring signatures [RST01]. In this section, we present a new kleptographic attack on CryptoNote's ring signature. This attack demonstrates how the uncontrolled randomness in ring signatures can be maliciously exploited to kleptographically leak the user's secret information, like their secret signing key. Note that the same principles apply to any other uncontrolled randomness in blockchain primitives.

As explained in Sec. 2.2.6.2, the CryptoNote protocol uses ring signatures. Mainly, it uses the ED25519 twisted Edwards curve, and the group order is a 253-bit prime p. The long term secret key of a user consists of two group elements $a, b \in \mathbb{Z}_p^*$. However, in practical implementations, a is commonly derived from b, where $a = \mathsf{hash}_p(b)$. Therefore, the long term secret key of a CryptoNote account is effectively 253 bits. As part of the one-time linkable ring signature, a *one-out-of-many* non-interactive zero-knowledge proof is included. More specifically, for a ring of size k, the format of the ring signature is

[2] `https://cryptonote.org/`

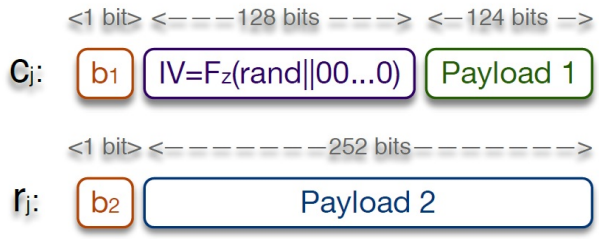


Figure 3.4: Generic kleptographic attack on CryptoNote: format of one pair of subverted random numbers (c_j, r_j) with 376 bits of leaked confidential information $\mathsf{sec} = (\mathsf{Payload\ 1} \| \mathsf{Payload\ 2})$. To ensure computational indistinguishability, the most significant bits, b_1 and b_2, are sampled according to the real distribution of c_j and r_j.

$\sigma = (I, c_1, \ldots, c_k, r_1, \ldots, r_k)$. Suppose the sender's public key is PK_ℓ , $\ell \in [k]$, then for all $j \in [k]$ and $j \neq \ell$, the components c_j and r_j are random group elements in $\mathbb{Z}_p$ and can be abused maliciously. Hence, our attack is premised on kleptographically exfiltrating the secret user's secret key in the ring signature's random numbers (c_j, r_j)'s.

In our attack, the attacker's encryption key is the same as the decryption key, which is a simple 128-bit random key denoted as z. This key is a shared secret between the subverted wallet and the attacker. In the following, we detail the proposed attack as a three-step process: **(1)** leaking the victim's secret information, **(2)** identifying subverted transactions, and **(3)** extracting the hidden secret. These steps are carried out by two parties: an oblivious victim sender called Alice and an attacker called Carol, who has distributed the subverted CryptoNote wallet.

Step 1: leaking Alice's secret information sec **(**Leak**).** The most significant bit of a random $\mathbb{Z}_p$ element does not have uniform distribution; it is more biased to 0. Hence, to ensure computational indistinguishability between the subverted random numbers, denoted by sr, and innocuous random elements $(c_j, r_j) \in \mathbb{Z}_p$, Alice's subverted wallet hides her secret information sec in the least significant 252 bits of c_j and r_j. The most significant bits, b_1 and b_2, are sampled according to the real distribution of c_j and r_j. As depicted in Fig. 3.4, the rest of the bits consist of a 128-bit IV, 124-bit $\mathsf{Payload\ 1}$, 252-bit $\mathsf{Payload\ 2}$. Let $F : \{0,1\}^{128} \times \{0,1\}^{128} \mapsto \{0,1\}^{128}$ be a block cipher that takes as input a 128-bit plaintext and a 128-bit key, and outputs a 128-bit pseudo-random ciphertext. Besides,

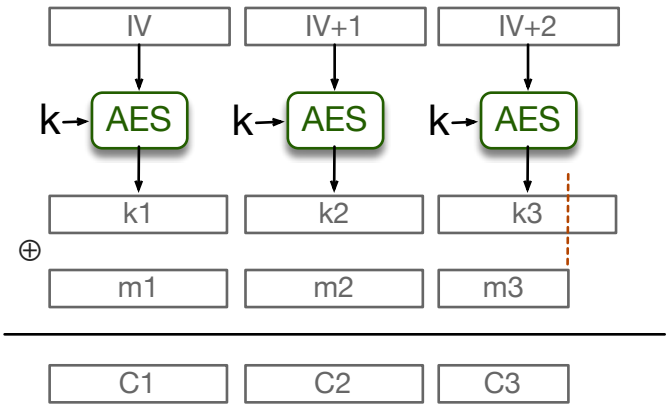


Figure 3.5: Ciphertext stealing CTS (CTR mode): CTS is used in our abstract kleptographic attack on CryptoNote because it ensures the length of the resulting ciphertext $(\mathsf{C1}, \mathsf{C2}, \mathsf{C3})$ is the same as the length of the plaintext $(\mathsf{m1}, \mathsf{m2}, \mathsf{m3})$.

Alice's wallet uses synthetic initial vector IV to allow Carol to efficiently identify which transactions on the blockchain contain Alice's subverted randomness sr. In particular, $\mathsf{IV} = F_z(\mathsf{rand} \| 00\ldots0)$, where rand is a 64-bits random string, and $00\ldots0$ is a 64-bit string of 0's. As a result, to check if a signature contains any sr, Carol can simply try to decrypt a suspected IV, obtaining $d = F_z^{-1}(\mathsf{IV})$. If the least significant half of d consists of 64 bits of 0's, then this signature contains sr.

In this attack, $\mathsf{Payload\ 1}$ and $\mathsf{Payload\ 2}$ are jointly used to convey a 376-bit hidden secret ($\mathsf{sec} = \mathsf{Payload\ 1} \| \mathsf{Payload\ 2}$). This can be used to leak Alice's 253-bit secret key s along with 123 extra bits, which can be used for integrity checks. The payloads are encrypted via a semantically secure symmetric encryption under Carol's secret key z and using IV. Also, to handle arbitrary-length hidden messages and ensure the resulting ciphertext has the same length as the message, the subverted wallet can use *Ciphertext Stealing* (CTS) [Dwo10]. Hence, our generic kleptographic attack on CryptoNote's ring signatures uses the ciphertext stealing technique in counter mode (CTR), where the last encryption block is truncated to fit the message length. For further clarity, the proposed encryption algorithm $\mathsf{CTS\text{-}Enc}_z(\mathsf{IV}, \mathsf{sec})$ is depicted in Fig. 3.5. More information on CTS and operation modes can be found in Dworkin's textbook [Dwo10].

Step 2: identifying signatures with subverted randomness sr. Before attempting to extract any leaked information from a subverted transaction, Carol should first identify

A Generic Kleptographic Attack on CryptoNote's Ring Signature

$\mathsf{KeyGen}(1^\lambda)$:

- Pick random: $z \xleftarrow{\$} \{0,1\}^{128}$;
- Return z;

$\mathsf{Leak}(z, \mathsf{sec})$:

- Pick random: $\mathsf{rand} \xleftarrow{\$} \{0,1\}^{64}$;
- $\mathsf{IV} = F_z(\mathsf{rand} \| 00\ldots0)$;
- $\hat{m} = \mathsf{CTS\text{-}Enc}_z(\mathsf{IV}, \mathsf{sec})$;
- $\mathsf{Payload}\ 1 = \hat{m}_{[0:123]}$;
- $\mathsf{Payload}\ 2 = \hat{m}_{[124:375]}$;
- Sample random: $c \xleftarrow{\$} \mathbb{Z}_p$, and $r \xleftarrow{\$} \mathbb{Z}_p$;
- $c_{[1:128]} = \mathsf{IV}$;
- $c_{[129:252]} = \mathsf{Payload}\ 1$;
- $r_{[1:252]} = \mathsf{Payload}\ 2$;
- Return (c, r);

$\mathsf{Extract}(z, (c, r))$:

- $\alpha = F_z^{-1}(c_{[1:128]})$;
- If $\alpha_{[64:127]} \neq (00\ldots0)$: Return $\perp$;
  Else: $\mathsf{IV} = c_{[1:128]}$;
- $\mathsf{Payload}\ 1 = c_{[129:252]}$;
- $\mathsf{Payload}\ 2 = r_{[1:252]}$;
- $\mathsf{sec} = \mathsf{CTS\text{-}Dec}_z(\mathsf{IV}, \mathsf{Payload}\ 1 \| \mathsf{Payload}\ 2)$;
- Return sec;

Figure 3.6: Pseudo-code for a generic kleptographic attack on CryptoNote's ring signature. The functions ($\mathsf{KeyGen}, \mathsf{Leak}, \mathsf{Extract}$) covertly exfiltrate a 376-bit secret sec by embedding it in *one* pair of innocuous-looking random numbers (c, r) of the ring signature, where the security parameter $\lambda = 128$.

if the transaction under inspection contains subverted randomness sr. To accomplish this, Carol parses IV from the first two c_j's of the ring signature σ in the transaction and checks whether the decryption of IV contains 64 bits of 0's, as shown in Fig. 3.4. Note that Leak exfiltrates the secret sec in one of the first two pairs of (c_j, r_j). If c_1 does not yield the identifying pattern, then Alice's secret index i must be 1, and Carol moves on to decrypt c_2, which must contain the IV; otherwise, the signature does not contain sr.

Step 3: extracting the hidden secret ($\mathsf{Extract}$). Once a subverted ring signature is successfully identified, Carol uses the $\mathsf{Extract}$ algorithm to extract Alice's secret information sec. Carol collects $\mathsf{Payload}\ 1$ and $\mathsf{Payload}\ 2$, as depicted in Fig. 3.4, and uses her secret key z to decrypt the payloads, obtaining $\mathsf{sec} = \mathsf{CTS\text{-}Dec}_z(\mathsf{IV}, \mathsf{Payload}\ 1 \| \mathsf{Payload}\ 2)$.

The pseudo-code in Fig. 3.6 further clarifies the generic kleptographic attack on CryptoNote's ring signature. Note that, in practice, the IV and Payload can be encrypted under two different keys derived from a single master key z. However, for notation simplicity, we use the same key here.

3.3.1 Security of kleptographic attack on ring signature

The security of the proposed kleptographic attack on CryptoNote is examined for undetectability under the security game in Fig. 3.3. Informally, if there is not any PPT watchdog $\mathcal{W}$ that can distinguish between the output of the subverted signing algorithm and that of the original, except for a negligible probability, then the proposed kleptographic attack is undetectable and is said to be secure.

Theorem 3.1. *If F and* CTS-Enc*, as shown in Fig. 3.6, are a secure pseudo-random function and a semantically secure encryption algorithm, respectively, then the kleptographic attack described in Fig. 3.6 is secure, i.e. undetectable by any PPT $\mathcal{W}$.*

Proof. We prove Theorem 3.1 using hybrid proofs and reduction. We show that if there is a PPT watchdog $\mathcal{W}$ that can detect our kleptographic attack on the ring signature with a non-negligible probability, i.e. $\mathcal{W}$ can distinguish between subverted signatures and normal signatures, then there must exist another PPT adversary $\mathcal{A}$ who can, with a non-negligible probability, break the PRF security of F or the semantic security of the encryption algorithm CTS-Enc. The security proof of Theorem 3.1 is presented in detail in Appendix C. □

3.4 Kleptographic Attacks on ECDSA

In the following subsections, we present two new kleptographic attacks on the ECDSA signature. The first is a stateful attack that mainly targets hardware wallets; it leaks the entire signer's secret key over two consecutive signatures. In comparison, the second attack is stateless with lower throughput and can be used to leak arbitrary information.

Kleptographic attack-1 on ECDSA: synthetic randomness

$\mathsf{Sign}^{(1)}(s, m_1)$:

- Pick random $r_1 \xleftarrow{\$} \mathbb{Z}_q$;
- Set $R' = g^{r_1} = (R'_x, R'_y)$;
- Set $w_1 = (\mathsf{hash}_p(m_1) + s \cdot R'_x) \cdot r_1^{-1} \pmod q$;
- Output $\sigma_1 = (R'_x, w_1)$;

$\mathsf{Sign}^{(2)}(s, m_2, r_1)$:

- Set $r_2 = \mathsf{hash}_p(Z^{r_1})$;
- Set $R = g^{r_2} = (R_x, R_y)$;
- Set $w_2 = (\mathsf{hash}_p(m_2) + s \cdot R_x) \cdot r_2^{-1} \pmod q$;
- Output $\sigma_2 = (R_x, w_2)$;

$\mathsf{Recover}(\sigma_1, \sigma_2, m_2, z)$:

- Parse $(R'_x, w_1) \leftarrow \sigma_1$, and $(R_x, w_2) \leftarrow \sigma_2$;
- Set $R' \leftarrow \mathsf{map}(R'_x)$;
- Set $r'_2 = \mathsf{hash}_p((R')^z)$;
- Output $s = (w_2 \cdot r'_2 - \mathsf{hash}_p(m_2)) \cdot (R_x)^{-1}$;

Figure 3.7: Kleptographic attack-1 on ECDSA: synthetic randomness. $\mathsf{Sign}^{(1)}$ is the same as the original ECDSA algorithm; however, it stores the ephemeral randomness r_1 in volatile memory. $\mathsf{Sign}^{(2)}$ generates r_2 using r_1 and the attacker's public key Z. r_2 is used by the attacker to recover the secret signing key s.

3.4.1 Kleptographic attack-1 on ECDSA: synthetic randomness

Our first proposed subversion attack on ECDSA leaks the secret key over two consecutive signatures. It depends on the synthetic-randomness generation of the second signature's ephemeral randomness r. Our attack can be viewed as a more succinct version of the attack proposed in [ME10]. In more details, let $z \in \mathbb{Z}_p$ be the adversary's secret key, and set the corresponding public key as $Z = g^z$. Let $R \leftarrow \mathsf{map}(R_x)$ be a mapping function that takes as input the x-coordinate of a point and outputs the corresponding point on the curve. To leak the entire signing key s over two consecutive signatures, the subverted ECDSA scheme runs two signing algorithms $\mathsf{Sign}^{(1)}$ and $\mathsf{Sign}^{(2)}$ consecutively. $\mathsf{Sign}^{(1)}$ is identical to the original signature algorithm; however, the subtle difference is that $\mathsf{Sign}^{(1)}$ stores the ephemeral key r_1 in a long-term memory, which can be accessed during the next signature invocation. $\mathsf{Sign}^{(2)}$ is also similar to the original signature algorithm except that it deterministically generates $r_2 = \mathsf{hash}_p(Z^{r_1})$, where Z is hardcoded in the subverted implementation.

Kleptographic attack-2 on ECDSA: rejection sampling

$\mathsf{Sign}(s, m_i, \mathsf{sec})$:

- Repeat the following process:
  - Pick random $r \xleftarrow{\$} \mathbb{Z}_q$;
  - Set $R = g^r = (R_x, R_y)$;
  - Compute $(j, b) \leftarrow \mathsf{PRF}_z(R_x)$;
  - If $\mathsf{sec}[j] = b$, break the loop;
- Set $w = (\mathsf{hash}_p(m_i) + s \cdot R_x) \cdot r^{-1} \pmod q$;
- Output $\sigma_i = (R_x, w)$;

$\mathsf{Recover}(\sigma_1, \ldots, \sigma_\ell, z)$:

- Init an array $\mathcal{S} = \emptyset$;
- For $i \in [\ell]$, do:
  - Parse σ_i as (R_{xi}, w_i);
  - Compute $(j_i, b_i) \leftarrow \mathsf{PRF}_z(R_{xi})$;
  - Set $\mathcal{S}[j_i] = b_i$;
- Output $\mathcal{S}$;

Figure 3.8: Kleptographic attack-2 on ECDSA: rejection sampling. The signatures are selectively chosen so that each signature leaks one bit b corresponding to the j^{th}-bit of the signer's secret information sec. The attacker can recover the leaked bits by intercepting the signatures and using the pseudo-random function PRF with his key z.

Once the adversary obtains two consecutive signatures σ_1 and σ_2, he can use his secret key z to recover the victim's signing key s. First, he parses σ_1 to (R'_x, w_1) and σ_2 to (R_x, w_2). Then, the attacker finds the point on the curve corresponding to R'_x by using $R' \leftarrow \mathsf{map}(R'_x)$. After that, the attacker computes $r'_2 = \mathsf{hash}_p((R')^z)$. Note, the attacker can verify the correctness of r'_2 by computing $R'' = g^{r'_2} = (R''_x, R''_y)$, and checking if R''_x is equal to R_x. The secret key can be extracted as $s = (w_2 \cdot r'_2 - \mathsf{hash}_p(m_2)) \cdot (R_x)^{-1}$. In this attack, the entire signing key s is leaked exclusively to the adversary over two signatures. For more clarity, the subverted ECDSA algorithm is shown in Fig. 3.7.

While this kleptographic attack has a high throughput, it has few drawbacks. First of all, it is stateful, so it is not suitable for every scenario, especially for software wallets. Furthermore, this attack can exclusively leak the signing key s and not arbitrary confidential information sec. Note, most cryptocurrency wallets can avoid the re-use of the address and signing keys. Thus, the leaked signing key in this attack may never be used again, even if the signing algorithm is executed twice with the same signing key.

Nevertheless, for most wallets, all the one-time signing keys are derived from a master key, which represents a more favourable target for attackers.

3.4.2 Kleptographic attack-2 on ECDSA: rejection sampling

To overcome the shortcomings of our first kleptographic attack on the ECDSA signature, our second attack on ECDSA is stateless and is designed to exfiltrate arbitrary information sec. As depicted in Fig. 3.8, the subverted signing algorithm takes as input the signing key s, the message m_i, and the secret $\mathsf{sec} \in \{0,1\}^n$ to be leaked. The signing algorithm leaks a random bit of sec per signature. Let $\mathsf{PRF} : \{0,1\}^* \times \{0,1\}^\lambda \mapsto \{0,1\}^{\log(n)} \times \{0,1\}$ be a pseudo-random function that takes as input an arbitrary-length message and the λ-bit PRF key, and outputs a random number of $\log(n+1)$ bits. The first $\log(n)$ bits is interpreted as an index j, and the last 1 bit is viewed as b. The subverted signing algorithm performs rejection sampling to find a random point $R = (R_x, R_y)$ such that $(j, b) \leftarrow \mathsf{PRF}_z(R_x)$ and $\mathsf{sec}[j] = b$. The rest of the signing process is identical to the original signature algorithm. Note that the rejection-sampling technique is efficient, and the expected number of repetitions per signature is 1.5 repetitions.

To recover the leaked secret sec, the adversary needs to obtain a collection of the signatures generated by the subverted algorithm. We emphasize that when the secret is a master key that can be tested for correctness, it is *not* necessary to leak the entire key in practice. Assuming the master key is 256 bits, to obtain 50% distinct key bits, the expected number of signatures is bounded by approximately 256 signatures. Asymptotically, to obtain n secret bits, we need $\theta(n \log n)$ signatures. The number of signatures needed to leak n secret bits can also be viewed as an instance of the coupon-collector problem [Fel57].

3.4.3 Security of attacks on ECDSA

In our first attack on ECDSA, the first signature's random number r_1 is picked randomly from $\mathbb{Z}_p$, and σ_1 is generated using the original algorithm. In comparison, the second signature's randomness is not picked randomly from $\mathbb{Z}_p$, but instead synthetically com-

puted using r_1 and the attacker's public key Z; $r_2 = \mathsf{hash}_p(Z^{r_1})$. Consequently, for any watchdog $\mathcal{W}$ inspecting the algorithm in black-box, i.e. $\mathcal{W}$ does not know Z, r_2 will be computationally-indistinguishable from elements randomly drawn from $\mathbb{Z}_p$. Also, in our second attack on ECDSA, the rejection-sampling process randomly draws r from $\mathbb{Z}_p$ in the same manner as the original algorithm, and the rest of the two algorithms, the subverted and original algorithms, are the same. Therefore, despite its intuitive longer execution time, our rejection-sampling attack on ECDSA has the same security as the original ECDSA algorithm. Hence, both of our kleptographic attacks on ECDSA are kleptographically secure, i.e. no PPT watchdog can distinguish with a non-negligible probability between the kleptographically-created signatures and normal signatures, as required by the security game in Fig. 3.3.

3.5 Kleptographic Attack Scenarios

As illustrated by our kleptographic attacks on CryptoNote's ring signature and ECDSA, the attacker can leak any of the victims' secret information by surreptitiously and maliciously manipulating the randomness within the generated signatures. The least that an attacker can achieve is to tag the generated transactions, which contain subverted signatures, to de-anonymize the senders. Although there could be various kleptographic scenarios, this section presents the attack scenario with the most severe consequences on the users of cryptocurrency wallets, which can result in stealing the users' funds. This scenario applies to open-source blockchain applications due to their complexity; however, it is more applicable to closed-source and hardware cryptocurrency wallets.

In this scenario, as in kleptography in general, the victimized sender is oblivious to the attack and can not detect it *in the black-box setting*. To steal the victims' funds, the saboteur implements an open-source cryptocurrency wallet with a kleptographic key-recovery mechanism. Since the attacker receives all broadcast transactions as the nature of blockchains entails, a hidden key-recovery attack is more plausible in blockchains as opposed to key recovery in other schemes, where the attacker needs to continuously spoof the network traffic, such as the attack on SSL/TLS and SSH2 by Goh et al. [GBPG03].

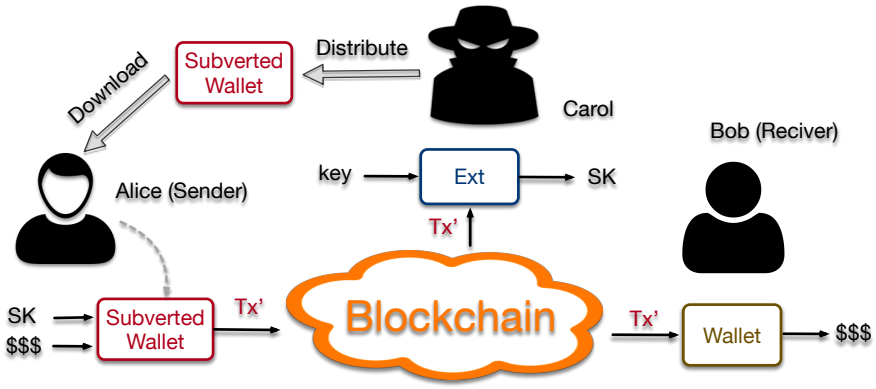


Figure 3.9: Kleptography attack: stealing victims' private signing keys

As depicted in Fig. 3.9, in this scenario, Alice is an innocent victim using a wallet that is maliciously implemented and distributed by a third party, Carol. In particular, Carol used a *kleptographic attack* to modify the wallet using one of the attacks explained in Sec. 3.3 and Sec. 3.4, so to leak the signer's private key, while evading detection in the black-box setting. Alice may have downloaded the sabotaged wallet, or her wallet may have been replaced by a sabotaged wallet using malware. Such malware infection is possible since, as reported by Kaspersky [Lab14], over one million bitcoin-wallet-stealing malware infections occur every month. In either case, whether downloaded by Alice or replaced by malware, whenever Alice posts any transaction to the blockchain using her subverted wallet, Carol can detect and extract the secretly leaked information as described in the previous sections. Once Carol has Alice's entire private key, she can impersonate Alice and steal all of her funds.

3.6 Realization of Kleptographic Attacks

This section demonstrates the realization of our kleptographic attack on CryptoNote's ring signatures, Sec. 3.3, on two real-world cryptocurrencies: Bytecoin and Monero. These two example cryptocurrencies have been chosen among other currencies for the following three reasons. (1) Both of these cryptocurrencies are based on the CryptoNote framework. (2) Their source code is publicly available. (3) We had access to some Monero coins that were essential for our demonstration to generate some transactions, and Bytecoin can

easily be mined to generate the needed coins. Besides, although Monero is based on the CryptoNote protocol, it uses the Borromean ring signature, which is different from the ring signature used in the CryptoNote protocol, as previously described in Sec. 2.2.6.2. Nevertheless, our generic attack in Sec. 3.3 is still applicable to Monero. Realizing the same kleptographic attack on Monero emphasizes that the same attack principles can be extended to all public blockchain applications with randomized cryptographic primitives.

In addition, this section demonstrates our attacks on the ECDSA scheme, as explained Sec. 3.4. We apply the two attacks on an implementation of the ECDSA signature rather than on any cryptocurrency that implements this scheme. We follow this approach because we do not have access to cryptocurrency coins for any of the cryptocurrencies that use ECDSA, such as Bitcoin. Also, other blockchain applications that may use the ECDSA signature do not offer open-source code nor proper or publicly-available documentation.

3.6.1 Kleptographic attack on Bytecoin

Bytecoin is an open-source cryptocurrency [Tea18] that has a market cap of around $42 millions[3]. Bytecoin follows the CryptoNote framework [Sab13], and hence uses the ED25519 twisted Edwards curve and CryptoNote's linkable ring signature to sign its transactions. As previously shown in Sec. 2.2.6.2, this signature has sufficiently many uncontrolled random numbers, which are highlighted in grey in Fig. 2.2. These random numbers could be exploited to communicate arbitrary information covertly.

Since Bytecoin closely follows the specifications of the CryptoNote framework, it can be directly attacked using the kleptographic attack described in Sec. 3.3. However, for code simplicity and clarity of demonstration, ciphertext stealing (CTS) is not used, and AES128 encryption is used because it is available in Bytecoin's source code. Also, as a proof of concept, instead of leaking a 376-bit secret sec, our subverted Bytecoin wallet leaks a shorter secret of 16 bytes. Similar to the description of the generic attack in Sec. 3.3, we have implemented our attack on the Bytecoin wallet (v 3.3.3) in the following

[3] `https://coinmarketcap.com/currencies/bytecoin-bcn/` on 12/July/2020.

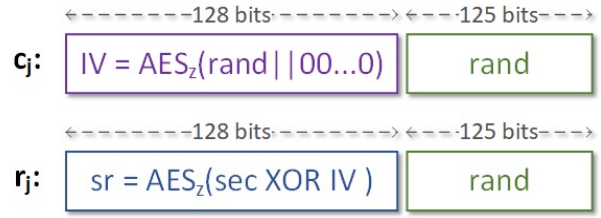


Figure 3.10: Bytecoin: leaking a 16-byte secret sec in one pair of (c_j, r_j) in the transaction's ring signature. If sec is used to indicate the signer's secret index, this pair (c_j, r_j) can be used as a preliminary packet. Hence, more secret information can be leaked in a single ring signature by embedding them in subsequent random numbers (c_j, r_j)'s where $j \neq \ell$ and ℓ is the signer's secret index.

three steps[4].

Step 1: secretly leaking a 16-byte secret sec and generating a signature that contains sr. To embed 16 bytes in a pair of random numbers (c_j, r_j), Alice's subverted wallet generates a synthetic initial vector; $\mathsf{IV} = \mathrm{AES}_z(\mathsf{rand} \| 00\ldots0)$, where rand is a 64-bit random string, $00\ldots0$ is a 64-bit string of 0's, and z is the attacker's symmetric key. Alice's wallet then places IV as the most significant 16 bytes of c_j and sets the rest of c_j randomly. After that, Alice's wallet uses this IV along with z to generate sr that is embedded in the most significant 16 bytes of r_j; $\mathsf{sr} = \mathrm{AES}_z(\mathsf{sec} \oplus \mathsf{IV})$. The format of (c_j, r_j) containing sr is illustrated in Fig. 3.10.

Furthermore, to implement this step of the attack, the Bytecoin wallet's source code is changed by mainly modifying one source file: crypto.cpp. The modified wallet simply alters the random numbers in the transaction's ring signature(s) by producing *one* pair of (c_j, r_j) as aforementioned. Note that $j \neq \ell$ where ℓ is the signer's *secret* index within the ring. The changes introduced to crypto.cpp affect the following two functions within the source file:

- generate_ring_signature(). This function is slightly modified to pass a counter to the random_scalar() function.
- random_scalar(). This function is modified by including an additional parameter in its input to specify the counter. When this counter is 0 and 1, random_scalar() generates c_j and r_j, respectively, which are subverted random numbers that hide

[4] Although the attack is described with respect to Bytecoin wallet (v 3.3.3), we confirm that the same principles are applicable to the latest release (v 3.5.1).

a 16-byte message, as depicted in Fig. 3.10. When the counter has different values, the function generates random scalars as per normal.

After generating the subverted signature, the transaction is sent as per usual over the blockchain. The attacker does not need to modify other parts of the wallet's source code.

Step 2: identifying signatures containing subverted randomness sr. To identify signatures containing sr, Carol checks every new transaction added to the ledger. To implement this step, the Bytecoin's wallet source file named BlockChainState.cpp is

Pseudo-code: kleptography attack on Bytecoin wallet

KeyGen(1^λ):
- Pick random $z \xleftarrow{\$} \{0,1\}^{128}$;
- Return z;

Leak$_z$(sec):

crypto.cpp:

generate_ring_signature():
- If$((j \neq \ell)\&(j == 0))$:
  - c_j = random_scalar(0);
  - r_j = random_scalar(1);
- Else: process as per normal;

random_scalar(n):
- rand $\xleftarrow{\$} \mathbb{Z}_p$;
- if$(n == 0)$:
  - IV = rand$_{[0:63]}$||zeros;
  - IV = AES$_z$(IV);
  - rand$_{[0:127]}$ = IV;
- if$(n == 1)$: rand$_{[0:127]}$ = AES$_z$(sec $\oplus$ IV);
- Return rand;

Extract$_z(c, r)$:

BlockChainState.cpp:

add_transaction(tx):
- for$(j = 0; j < 2; j{+}{+})$
  - IV$'$ = AES$_z^{-1}(c_{j,[0:127]})$;
  - if(IV$'_{[64:127]}$ == zeros):
    - sec = AES$_z^{-1}(r_{j,[0:127]}) \oplus c_{j,[0:127]}$;
    - Return sec;
- Return 0; % *No hidden message*

Figure 3.11: Pseudo-code for the implementation of a kleptographic attack on the Bytecoin wallet. Two files are slightly changed crypto.cpp and BlockChainState.cpp to leak a 16-byte secret sec. z is the adversary's secret key, and ℓ is the signer's index within the ring signature. The security parameter $\lambda = 128$.

Demo Bytecoin Transaction

- **Block height:** 1671177
- **Transaction id:** 52caba6ef4e4716ac8a25681eb3f380d3d1fee057ada7eb62d687a f36f1a44ff
- **Sender's address:** 26c6YmZmLJZYxnVAt56kRraBhxiEUt8yoJR3VV4UV5 VcRM9Pzs5qV7KStQHaa7xkAHej3WTTxtAc1KHbCSPoZ2ms3bdUsY6.
- **Receiver's address:** 26c6YmZmLJZYxnVAt56kRraBhxiEUt8yoJR3VV4U V5VcRM9Pzs5qV7KStQHaa7xkAHej3WTTxtAc1KHbCSPoZ2ms3bdUsY6.
- **Mixin count (ring size):** 6

Figure 3.12: An example of a kleptographically-generated Bytecoin transaction.

slightly modified to check each signature by decrypting each pair of (c_j, r_j) numbers. Carol uses her key z to decrypt the most significant 16 bytes of c_j to check if it contains 64 bits of zeros, as in Fig. 3.10. If she detects such a pattern, Carol identifies the existence of subverted randomness and sets IV as the most significant 16 bytes of c_j. If, however, no such pattern is detected, then the signature does not contain any subverted randomness.

Step 3: extracting hidden secret sec. After identifying the existence of sr, Carol decrypts the most significant 16 bytes of r_j to extract sec, that is $\text{sec} = \text{AES}_z^{-1}(r_{j,[0:127]}) \oplus (c_{j,[0:127]})$.

To further demonstrate our kleptographic attack on Bytecoin's wallet, Fig. 3.11 provides a pseudo-code describing the implementation of the attack on Bytecoin. Also, Fig. 3.12 presents a demo transaction included in the block at height 1671177 that contains a 16-byte hidden message. To extract and verify the existence of the kleptographically-leaked information, we provide a tool that can be downloaded from our GitHub repository[5]. The repository also contains the actual transaction binary in tx.txt and a pair (c, r) of random numbers containing a 16-byte secret in cr.txt. The transaction hash in Fig. 3.12 can be seen in any Bytecoin explorer[6], and the provided transaction binary should hash to the same hash value.

Note that the same attack can be easily extended to leak longer secret information. Notably, the subverted randomness sr in the first pair of (c_j, r_j), as shown in Fig. 3.10, can be used as a preliminary packet that specifies the length of the leaked information and

[5] `https://github.com/NaLancaster/hash_and_extract`
[6] An example of such explorers is: `https://minergate.com/blockchain/bcn/blocks`

the signer's index within the ring. Hence, more information can be leaked in subsequent subverted random numbers (c_j, r_j)'s, where $j \neq \ell$ and ℓ is the signer's secret index.

3.6.2 Kleptographic attack on Monero

Monero is a cryptocurrency that has a market cap of over \$1 billion[7]. Similar to Bytecoin, Monero is based on the CryptoNote framework; however, it has had adopted the use of the Borromean ring signature [MP15] explained in Sec. 2.2.6.2, instead of CryptoNote's ring signature. Our demonstration kleptographic attack on Monero exploits the random numbers within the Borromean ring signature. Note that as of October 2018, Monero (v 0.13.0.0) has replaced Borromean ring signatures, that are exploited by our attack, by succinct zero-knowledge proofs called Bulletproofs [BBB+18], which are not covered by this work. Consequently, all of our discussion concerning Monero is regarding previous versions of the source code, (v 0.12.0.0) and older, that use Borromean ring signatures.

Monero has a very complex cryptographic structure and ring signature scheme in particular. The core of Monero's wallet involves the Multilayered Linkable Spontaneous Anonymous Group Signature (MLSAG) and the Borromean ring signature [MP15]. MLSAG is similar to the 1-out-of-n ring signature that is used as part of the CryptoNote protocol; however, rather than using a ring signature on a set of n keys, MLSAG uses a ring signature on a set of n-key vectors. Using MLSAG, the signer proves to know all the private keys corresponding to one column in the public keys' matrix. Despite the massive one-time secret key, the long-term secret key is still a single group element in $\mathbb{Z}_p$. Borromean ring signature [MP15], which is a generalization based on the 1-out-of-n signature [AOS02], is used to mask the transferred amount while enabling the receiver to know how much they have received by revealing the mask [Max].

In our experiment, we chose to exploit the Borromean ring signature because it offers higher throughput and is similar to CryptoNote's ring signature. Nevertheless, though with lower throughput, different primitives could also be exploited to mount kleptographic attacks. Although longer secrets can be leaked, for simplicity, our demonstration attack

[7] `https://coinmarketcap.com/currencies/monero/` on 12/July/2020.

Figure 3.13: Monero: exfiltrating a 32-byte secret in a pair of subverted random numbers within the Borromean ring signature. $s_{0,1}$ contains IV, the index of $s_{0,2}$ within the ring, and an identification pattern of 15 zeros. $s_{0,2}$ contains the leaked information $\mathsf{sec} = (\mathsf{sec}_1||\mathsf{sec}_2)$.

on Monero leaks a 32-byte secret sec in the randomly generated $s_{i,j}$ numbers as part of the Borromean ring signature [MP15].

Specifically, two vectors of $s_{i,j}$ numbers are generated by the genBorromean() function: $s_{0,j}$ and $s_{1,j}$. Besides, $s_{0,j}$'s are randomly generated when the j^{th} bit commitment is 1. Two of these randomly generated $s_{0,j}$'s are used to hide sec, as shown in Fig. 3.13. In a similar manner to our attack on Bytecoin, we use AES128 encryption because it is already available in the source code. The attack is mounted on Monero's wallet according to the following three steps between a sender, Alice, who is unknowingly using a wallet that is kleptographically implemented by an attacker, Carol.

Step 1: leaking a 32-byte secret sec and generating a signature that contains sr. Alice's wallet is modified to covertly exfiltrate a 32-byte secret sec in the randomly generated $s_{i,j}$ numbers of the Borromean ring signature. In particular, the subverted randomness sr containing the secret string sec is embedded in two $s_{0,j}$ numbers, where the j^{th} bit commitment is 1. For simplicity, we use $s_{0,1}$ and $s_{0,2}$ to denote the first two randomly generated numbers in the $s_{0,j}$ vector, although they might not necessarily correspond to $j = 1$ and $j = 2$, respectively.

Fig. 3.13 shows the two subverted numbers. The first number, $s_{0,1}$, includes 16 bytes of random IV concatenated with 1 byte representing the index of $s_{0,2}$ and 15 bytes of zeroes. The least significant 16 bytes of $s_{0,1}$ are encrypted using AES128 in the CBC mode. The second subverted random number, $s_{0,2}$, contains the leaked secret sec encrypted using AES128 under Carol's key z.

This step of the attack is achieved by slightly modifying two functions: genBorromean() and skGen() in two source files: rctSig.cpp and rctOps.cpp, respectively. genBorromean()

Demo Monero Transaction

- **Block height:** 1502164
- **Transaction id:** $e4b7982b081a17892525f1b1d3011ec06a0820cbf451d3a64f8ea998104a753c$
- **Sender's address:** $455Bu1zXzgXEeXxrjzRSsEifP8WgtLTKYLreQ7RrA1fcFi2UKjgtc2UBapB9AcDaitdY7SdWGFsEZRELL8A1nMnEFRVZg47.$
- **Receiver's address:** $42F5itWciYAg5QJxZEqWz5hrQNFaySUbxfxsjcdp8FnrRM68c8Nzujm3UqfscVC6r2c2GwuiP4sRsQv3ZZUc1spjUHuDHsx.$
- **Mixin count (ring size):** 5

Figure 3.14: An example of a kleptographically-generated Monero transaction

is modified to pass two extra parameters to skGen(). The first parameter is a counter that indicates which of the two random numbers is to be generated. The second parameter represents the index of the j^{th} bit corresponding to the position of the second number $s_{0,2}$ within the $s_{0,j}$ vector. When the value of the counter is 0 or 1, skGen() generates random numbers according to Fig. 3.13, and executes as normal otherwise.

Step 2: identifying signatures containing subverted randomness sr. To identify transactions containing sr, the source file blockchain.cpp is modified to check the randomness within each new transaction and identify subverted signatures. Carol tests each number in the $s_{0,j}$ vector by looking for a random IV that decrypts the second half of the tested number to a similar pattern to $s_{0,1}$ in Fig. 3.13. Once this pattern is detected, Carol concludes that this signature contains sr and retrieves the index of $s_{0,2}$ from the 16^{th} byte of $s_{0,1}$.

Step 3: extracting the leaked secret sec. When a subverted signature is detected, Carol retrieves the index of $s_{0,2}$. Then, she extracts the hidden secret by decrypting $s_{0,2}$ using her key z with AES128. As a further demonstration, Fig. 3.14 shows a Monero transaction that has been subverted according to this attack and has been successfully posted to the Monero blockchain.

3.6.3 Demonstration of kleptography on ECDSA

Lacking access to the needed coins for any of the cryptocurrencies that use the ECDSA signatures, and not being able to find easy-to-read open-source ECDSA-based blockchain

Number of Signatures to Leak Key Bits							
Exp. #	**32**	**64**	**96**	**128**	**160**	**192**	**224**
1	35	71	117	173	227	314	520
2	33	73	119	189	253	344	485
3	32	74	120	170	230	339	471
4	34	70	114	179	238	335	491
5	32	69	119	189	280	385	552
6	32	76	122	170	233	333	526
7	36	76	120	180	262	400	576
8	32	71	127	197	259	368	566
9	33	72	115	162	242	348	528
10	32	71	121	177	243	363	498
11	31	70	121	180	246	345	524
12	35	76	120	181	260	386	563
13	33	69	124	190	251	352	506
14	32	72	121	180	255	353	518
15	33	78	124	178	246	355	539
16	34	72	113	168	232	331	522
17	35	72	111	162	228	340	512
18	31	79	125	201	281	378	544
19	37	76	122	174	243	329	475
20	34	75	121	175	260	369	570
Average	33.3	73.1	119.8	178.8	248.4	353.4	524.3
Std. dev.	1.6	2.9	3.9	10.2	15.2	21.6	30.5

Table 3.1: Number of signatures needed to obtain $32, 64, 96, 128, 160, 192$, and 224 bits out of the total 256 key bits. This experiment was run 20 times to record the number of needed signatures to leak some bits of the secret key. As seen in this table, the average number of signatures that should be intercepted by the attacker to retrieve 50% of the key, i.e. 128 bits, is about 179 signatures.

applications, we have chosen to demonstrate our two kleptographic attacks on ECDSA directly on an implementation of the signature. As such, we have successfully implemented both attacks from Sec. 3.4 on an open-source ECDSA library [ecd14]. The source code of the kleptographically-implemented ECDSA scheme with both attacks can be found in our public repository in GitHub[8].

As demonstrated by the source code, when using the synthetic-randomness attack described in Sec. 3.4.1, the adversary leaks the entire signer's key over two consecutive signatures. Also, using the same source code, we have demonstrated the efficiency of the rejection-sampling attack on ECDSA by experimenting with the needed number of signatures to leak $32, 64, 96, 128, 160, 192$, and 224 bits out of the total 256 secret key bits s. This experiment was run 20 times to record the number of needed signatures to

[8] `https://github.com/NaLancaster/ecdsa_klepto_attacks`

leak some bits of the secret key. As shown in Table 3.1, the average number of signatures that should be intercepted by an attacker to retrieve 50% of the key, i.e. 128 bits, is about 179 signatures, which confirms the theoretical computation bounds explained in Sec. 3.4.2.

3.7 Summary

In this chapter, we described the notion of kleptography and its plausibility in public blockchains. After that, we proposed three new kleptographic attacks on two commonly used signature schemes in blockchains: ring signature and ECDSA. We also explained the attack scenario with the most severe consequences, which is to implement kleptographic key-recovery schemes in cryptocurrency wallets. This scenario is dangerous as it leads to secretly stealing the users' signing keys and eventually stealing their cryptocurrency funds. All of the three attacks are *passive*, where the attacker does not need to interact with the subverted wallets actively, *interoperable* since modified wallets transact seamlessly with normal wallets, and *undetectable* in the black-box setting. Moreover, we described how we had realized our kleptographic attack on the ring signature in two real-world cryptocurrencies: Bytecoin and Monero. Finally, due to the lack of access to any crypto coins of any well-documented open-source cryptocurrency that uses the ECDSA signature, such as Bitcoin, we demonstrated both of our kleptographic attacks on ECDSA on an open-source ECDSA implementation rather than on a real cryptocurrency.

Chapter 4

Steganography in Blockchains

In the previous chapter, we demonstrated the susceptibility of randomized signatures in public blockchains to kleptographic attacks, which can lead to mass-scale stealing of users' funds in cryptocurrencies. In this chapter, we show that *steganography* represents another plausible threat to public blockchains. Sec. 4.1 introduces the notion of steganography, its security definitions, and its properties. After that, in Sec. 4.2, we explain the current state of steganography in blockchains, clarify the relationship between undetectable kleptographic attacks and secure steganographic communication, and discuss the repercussions of steganography abuse on the users and on the blockchain technology itself. Then, Sec. 4.3 demonstrates a scenario in which public blockchains are abused for covert broadcast communication. In particular, we design, implement, and evaluate a broadcast communication tool called *Skywhisper* on top of a real-world cryptocurrency. Besides, we show that Skywhisper presents a robust and provably-secure steganographic broadcast channel.

4.1 Steganography

The concept of *steganography* was introduced by Simmons' *prisoner's problem* [Sim84]. Steganography refers to the techniques that allow the covert transmission of a message over a communication channel so that the mere presence of the hidden message is not detectable to a warden who monitors the channel [HvL09, DIRR09]. Modern steganog-

raphy techniques can be applied to various media, such as images, audio, HTML files, etc. [PAK99].

More formally, let $\mathcal{C}$ be a channel on the alphabet Σ with length ℓ, which can be viewed as a function that maps the channel history $\mathcal{H} \in (\Sigma^{\leq \ell})^*$ to a probability distribution upon $\Sigma^{\leq \ell}$. Denote the probability distribution of a given channel by $\mathcal{C}_{\mathcal{H}}$. A stegosystem on a family of channels $\mathbf{C} = \{\mathcal{C}^\lambda\}_{\lambda \in \mathbb{N}}$ consists of three PPT algorithms $\mathcal{ST} = (\mathsf{KeyGen}, \mathsf{Embed}, \mathsf{Extract})$ as follows:

- $(\mathsf{ek}, \mathsf{dk}) \leftarrow \mathsf{KeyGen}(1^\lambda)$ is a key generation algorithm that takes as input a security parameter 1^λ, and outputs an embedding key ek and an extraction key dk.
- $\mathsf{st} \leftarrow \mathsf{Embed}^{\mathcal{C}^\lambda_{\mathcal{H}}}_{\mathsf{ek}}(m, \mathcal{H})$. Given an embedding key ek, a message m, and a channel history $\mathcal{H}$, Embed generates a stegotext message st. Note that Embed has sampling access to $\mathcal{C}^\lambda_{\mathcal{H}}$.
- $m \leftarrow \mathsf{Extract}_{\mathsf{dk}}(\mathsf{st})$. $\mathsf{Extract}$ takes as input an extraction key dk and a stegotext st, and outputs the hidden message $m \in \{0,1\}^*$.

Stegosystem security. The stegosystem's goal is to communicate a hidden message covertly by concealing the mere existence of the secret communication. Therefore, a stegosystem, denoted as $\mathcal{ST}$, is considered to be *computationally secure* if a computationally-bounded observer or warden, who knows the channel history $\mathcal{H}$

Stegosystem CHA-Experiment

$\mathbf{Expt}^{\mathsf{CHA}}_{\mathcal{A}}(1^\lambda)$

1. $\mathcal{A}(1^\lambda)$ outputs a message m;
2. $\mathcal{O}$: $(\mathsf{ek}, \mathsf{dk}) \leftarrow \mathsf{KeyGen}(1^\lambda)$;
3. $\mathcal{O}$: $b \xleftarrow{\$} \{0,1\}$;
4. **If** $b = 0$: $\mathsf{st} \leftarrow \mathsf{Embed}^{\mathcal{C}^\lambda_{\mathcal{H}}}_{\mathsf{ek}}(m, \mathcal{H})$;
   Else: $\mathsf{st} \xleftarrow{\$} \mathcal{C}^\lambda_{\mathcal{H}}$;
5. $\mathcal{O}$ passes st to $\mathcal{A}$;
6. $\mathcal{A}(c)$ outputs a bit b^*;
7. $\mathcal{O}$: returns $b \stackrel{?}{=} b^*$;

Figure 4.1: Stegosystem CHA-security experiment between a challenger oracle $\mathcal{O}$ and an adversary $\mathcal{A}$. $\mathcal{O}$ returns $\mathsf{st} \leftarrow \mathsf{Embed}^{\mathcal{C}^\lambda_{\mathcal{H}}}_{\mathsf{ek}}(m, \mathcal{H})$ if $b = 0$ and returns $\mathsf{st} \xleftarrow{\$} \mathcal{C}^\lambda_h$ if $b = 1$.

and the hidden text m, is not able to distinguish the stegotext st from objects randomly picked from the channel distribution $\mathcal{D}$ [HvL09]. The security of a stegosystem is defined by a *chosen hidden-text attack* (CHA) security experiment, as shown in Fig. 4.1. According to this security experiment, we say a stegosystem $\mathcal{ST} = (\mathsf{KeyGen}, \mathsf{Embed}, \mathsf{Extract})$ is CHA-secure if the advantage of an adversary $\mathsf{Adv}^{\mathsf{CHA}}_{\mathcal{A},\mathcal{ST}}$ in distinguishing a steganographic text $\mathsf{st} \leftarrow \mathsf{Embed}^{\mathcal{C}^\lambda_{\mathcal{H}}}_{\mathsf{ek}}(m, \mathcal{H})$ from innocent random text $\mathsf{st} \xleftarrow{\$} \mathcal{C}^\lambda_h$ is negligible. In other words, $\mathcal{ST}$ is CHA-secure if there is not any PPT adversary who can win this security experiment with a non-negligible probability.

Definition 4.1. *Based on the security experiment shown in Fig. 4.1, we say a stegosystem $\mathcal{ST}$ is indistinguishable chosen-hidden text-attack (IND-CHA) secure if the advantage for all PPT adversary $\mathcal{A}$ in winning the experiment, denoted as $\mathsf{Adv}^{\mathsf{CHA}}_{\mathcal{A},\mathcal{ST}}$, is negligible, i.e. the following statement is valid:*

$$\mathsf{Adv}^{\mathsf{CHA}}_{\mathcal{A},\mathcal{ST}}(1^\lambda) = \left| \Pr \left[\begin{array}{l} (\mathsf{ek}, \mathsf{dk}) \leftarrow \mathsf{KeyGen}(1^\lambda); b \leftarrow \{0,1\}; \\ b^* \leftarrow \mathcal{A}^{\mathcal{O}(b,m,\mathcal{H})}(1^\lambda) : b = b^* \end{array} \right] - \frac{1}{2} \right| \leq \mathsf{negl}(\lambda)$$

$\mathcal{O}(b, m, \mathcal{H})$ is an oracle that returns $\mathsf{st} \leftarrow \mathsf{Embed}^{\mathcal{C}^\lambda_{\mathcal{H}}}_{\mathsf{ek}}(m, \mathcal{H})$ if $b = 0$ and returns $\mathsf{st} \xleftarrow{\$} \mathcal{C}^\lambda_h$ if $b = 1$.

Stegosystem correctness, efficiency, and robustness. For a stegosystem $\mathcal{ST}$ to be useful, it has to be *correct*. Correctness implies that if Embed is used to hide a message m with the key ek and channel history $\mathcal{H}$, then, except for a negligible probability $\mathsf{negl}(\lambda)$, $\mathsf{Extract}$ should be able to retrieve the hidden message m given the key dk and the corresponding stegotext st [HvL09].

Definition 4.2 (Correctness)**.** *We say a stegosystem $\mathcal{ST} = (\mathsf{KeyGen}, \mathsf{Embed}, \mathsf{Extract})$ is correct if for all $(\mathsf{ek}, \mathsf{dk}) \leftarrow \mathsf{KeyGen}(1^\lambda)$ we have:* $\Pr\left[m \leftarrow \mathsf{Extract}_{\mathsf{dk}}(\mathsf{Embed}^{\mathcal{C}^\lambda_{\mathcal{H}}}_{\mathsf{ek}}(m, \mathcal{H}))\right] \geq 1 - \mathsf{negl}(\lambda)$.

Besides security and correctness, the following two properties are also important for stegosystems. (1) Reliability/efficiency indicates the probability that an embedded

message is extracted when the stegosystem does not achieve perfect correctness. (2) Robustness measures the inability of a warden to alter the sender's communication transcript, that contains the hidden message(s), and prevent the receiver from recovering the hidden message(s). For more definitions of steganography, please refer to [HvL09, DIRR09].

4.2 Steganography in Blockchains

Although not focused on steganographically embedded content, there have been multiple media reports of arbitrary objectionable content that is found stored in blockchains [S. 18, H. 18, BBC19, Sky18]. Besides these reports, there has been a significant body of academic research on this topic. For example, Matzutt et al. reported that 0.8% of 146 million Bitcoin transactions store content on the blockchain or use non-standard scripts [MHH+16]. Later, Matzutt et al. surveyed the methods that are used to store non-financial content and discovered that 1.4% of *all* Bitcoin transactions contain non-financial data. They also retrieved over 1600 files, some of which contain immoral content [MHH+18].

Content insertion in public blockchains has enabled the realization of some new innovative applications [FHBS19], such as censorship-circumvention tools [AZ19b, RC19], consensus agreement [TD17], pseudonymous identities [FWB15], name services[1], and timestamping[2]. For example, Tithonus [RC19] and Catena [TD17] are censorship-circumvention and consensus agreement applications, respectively, that depend on inserting data in Bitcoin transactions. Also, R3C3 is a censorship-bootstrapping tool that is built on top of Zcash [MMSK18]. However, although the absence of a central censor makes blockchains appealing in some use cases, the increasing amount of illicit content posted to the blockchains poses a regulatory challenge [STW+16]. Also, as reported by Matzutt et al. [MHH+18], some countries may prosecute individuals for the mere possession of specific digital content.

Subsequently, to deter the insertion of arbitrary content in blockchains, several

[1] `https://www.namecoin.org/`
[2] `https://opentimestamps.org/`

techniques have been discussed to either filter out unwanted content before it is added to the ledger [MHZ+18] or remove content from the blockchain [PDC17, AMVA17]. However, all of the proposed countermeasures can only be effective if the malicious content attached to the transactions is detectable. Naively, one can attempt to avoid detection by encrypting the malicious content and attaching its ciphertext to a transaction. However, it is noticeable to the public that there is suspicious data attached. Nonetheless, as seen in the rest of this chapter, detection is not feasible if data is steganographically hidden into standard transactions in blockchains.

4.2.1 Steganography from kleptographic attacks

The relationship between kleptography and steganography is well known. For example, Russell et al. described techniques to destroy steganographic channels in randomized algorithms and consequently deter kleptographic attacks [RTYZ16b]. Nonetheless, Berndt and Liśkiewicz were the first to formally prove the connection between kleptography and steganography. They concluded that undetectable kleptographic attacks correspond to secure stegosystems on certain cryptographic primitives and vice versa [BL17].

Motivated by the proven correspondence between kleptography and steganography [BL17], in this chapter, we demonstrate how the kleptographic attacks presented in Chapter 3 can be used to realize steganographic systems in public blockchains. However, it is worth noting that due to their different attack assumptions, a protection mechanism may be efficient against kleptography and totally ineffective against steganography, as shall be seen in more detail in Chapter 5 and Chapter 6. This difference necessitates distinguishing between steganography and kleptography and justifies the separate treatment of the two types of attacks on blockchains.

Chapter 3 clarified the plausibility of kleptographic attacks on blockchains in the black-box setting, and attributed their significance and plausibility to eight reasons: (1) blockchains' complex software, (2) the adversaries mis-implementations of well-established schemes to curtail their security, (3) the counter-intuitive highly centralized development of many blockchain applications, (4) the inability of the end-users to inspect

(a) Kleptography: the watchdog scrutinizes the subverted wallet in the black-box settings, and examines all generated transactions.

(b) Steganography: the user is willingly participating in the attack, and the warden monitors the channel.

Figure 4.2: The main informal difference between kleptography and steganography is the role of the end-user, who is an oblivious victim in the former and a complicit party in the latter.

the authenticity of open-source blockchain applications, (5) the ease of distributing kleptographically-modified blockchain applications, (6) the broadcast nature of blockchains, (7) the secure anonymity techniques, which help the attackers avoid de-anonymization, and (8) the detrimental consequences on the victims' funds and privacy. By comparison, steganography is even more plausible due to its different assumptions. In kleptography, the sender, Alice, is an oblivious victim of the attack and should not be able to detect any discrepancies when executing her subverted wallet. In contrast, in steganography, Alice is complicit, and the attack is considered successful if the crafted transactions carrying the hidden information are undetectable by any PPT watchdog scrutinizing the channel rather than the wallet itself. In this sense, *informally*, the difference between kleptography and steganography in blockchains can be viewed as the difference in the complicity of the end-user, as depicted in Fig. 4.2.

As such, the kleptographic attacks on CryptoNote's ring signature and ECDSA described in Sec. 3.3 and Sec. 3.4 can be used to realize secure steganographic systems in blockchains. However, due to the different assumptions, the attack scenarios differ from those in Chapter 3. Also, and as a result of the different assumptions, the corresponding countermeasures are different, as shall be discussed later in Chapter 5 and Chapter 6.

Despite the well-known relationship between kleptographic and steganographic attacks, before our work, there had been only one suggestion to mis-implement blockchains to establish steganographic communication [Par18]. Partala presented a proof-of-concept attack on Bitcoin to steganographically embed one bit into the standard Bitcoin trans-

action's recipient address without being distinguished from an innocuous transaction and without burning any funds [Par18]. According to their attack, to covertly send a hidden message m, the sender sends $\alpha = \delta||m'$, where m' is the symmetric cipher of m encrypted under the key k, i.e. $m' = \text{Enc}_k(m)$, and δ is a starting pattern known to both the sender and the receiver(s). In addition, the string α is sent bit by bit through the rejection-sampling of the transaction address a. The least significant bit of the selectively chosen address a is the same as the i^{th} bit of α, i.e. $a[\mathsf{LSB}] = \alpha[i]$. Only one bit is sent in every transaction, and, to maintain order, only one transaction is sent in each block. Therefore, with 10 minutes to add a new block in Bitcoin, the sender needs more than 24 hours to send a message of 20 bytes. Moreover, the receiver continuously checks the payments generated by the sender, and if he ever detects δ, he retrieves m' bit by bit and decrypts it to reveal m; $m = \text{Dec}_k(m')$. Notably, because the receiver needs to know the sender's identity, this method does not apply to blockchains where the sender's identity is anonymous, e.g. Zcash.

Our work in this chapter extends the study of steganography in blockchains in two ways. Firstly, it presents a more practical and higher-throughput steganographic communication in blockchains in Sec. 4.3. Secondly, it practically demonstrates the correspondence between undetectable kleptographic attacks on signatures and the existence of secure steganographic channels in these signatures.

4.2.2 Repercussions of steganography on blockchains

It remains to clarify the actual adverse consequences of exploiting public blockchains to establish steganographic communication on both the users and the technology itself. Malicious users who communicate objectionable data do not only pose a potential ethical threat but may also hinder the future of public blockchains. This is due to the fact that *most* public blockchains permanently store their transactions and store replicas in the host machines of all the participating P2P nodes. Hence, if any of the transactions contain malicious content, all participating users will have the same content in their machines. Thus, from a moral standpoint, users may refrain from participating in such

blockchains, which may jeopardize the future adoption of the blockchain technology.

Although the immutability of the ledger is seen as a necessary feature for many applications, such as cryptocurrencies, it contradicts the users' *right to be forgotten* (RtbF) defined by the European General Data Protection Regulation (GDPR) [Eur16, PCAP19]. The RtbF entails that users should be able to delete their own digital personal records when certain conditions are met. On the contrary, if any personal information is uploaded to the blockchain, it will not be technically possible to erase the information without a hard fork of the chain [PDC17]. Likewise, inaccurate data that is generated by a corrupted user-side application, e.g. a modified cryptocurrency wallet, and added to the blockchain will remain inaccurate until corrected by a hard fork [ZBA18]. The situation is exacerbated if the information represents immoral content [MHH+16, MHH+18]. For example, Matzutt et al. listed some of the arbitrary content that has been found in blockchains: '(1) copyright violations, (2) malware, (3) privacy violations, (4) politically-sensitive content, and (5) illegal content' [MHH+18].

Furthermore, unlike other systems, such as social media platforms, blockchains store data irrevocably, and all blockchain data is downloaded and locally stored by the users. Therefore, if a public blockchain is known to the authorities to be abused for covert communication or storage of malicious content, then authorities in any given country may criminalize the mere participation in such blockchains [MHH+18]. As such, out of fear of prosecution, many users may resort to the following two options. The first possible option is that users may simply choose not to participate in public blockchains, which will obstruct the adoption of immutable public blockchains. The second option is that users may choose not to store the full ledger, which defeats the purpose of decentralized blockchains. This behaviour leads to a more centralized setting, where fewer users actively participate in the consensus protocol. Besides, this behaviour may facilitate Sybil attacks [Dou02], where malicious parties use multiple identities to control more power in the consensus protocol, which will lead to a biased consensus that can be used to mount double-spending attacks, denial-of-service attacks, and privacy attacks [CSLR18].

In short, steganography has adversarial effects on public blockchains as it allows the

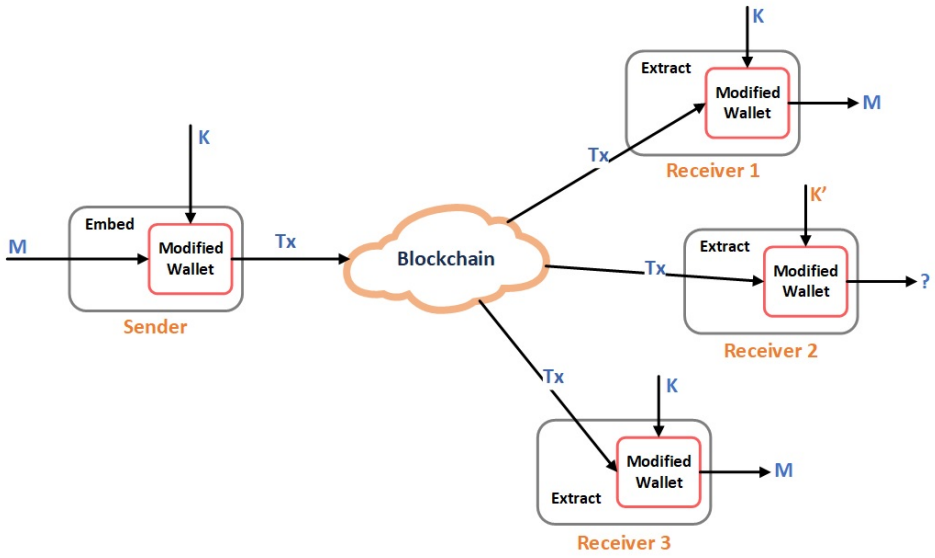


Figure 4.3: Covert broadcast communication on public blockchains. Only the receivers who have the broadcast encryption key K can detect broadcast-communication transactions and successfully extract the message M.

abuse of blockchains to store and disseminate arbitrary objectionable content. This abuse will inevitably lead to the criminalization of public blockchains that are known to the authority to store such content. Besides, steganographically-embedded content may lead users to abandon public blockchains, which will endanger the future adoption of the technology.

4.2.3 Steganographic attack scenarios

Besides its above-mentioned threat to the users and the technology itself, steganography can be used in blockchains in two particular scenarios: **(i)** to establish covert broadcast communication, and **(ii)** to secretly and persistently store arbitrary content.

Steganography scenario 1: covert broadcast channel. Conventional steganographic techniques typically assume that covert communication is between two parties: a sender and a receiver. On the contrary, we demonstrate in Sec. 4.3 how public blockchains can be abused to realize covert broadcast communication, i.e. one sender and multiple receivers, as depicted in Fig. 4.3. We show how a single Bytecoin transaction of 4 inputs and 10 public keys can steganographically transmit more than 2KB and costs about $0.0000215, given the price of a single Bytecoin coin (BCN) is $0.000215, and

the minimum transaction fee is 0.1 BCN[3]. The feasibility of this attack and the high throughput demonstrate the severity of this scenario, especially if abused by outlaws for illicit communication.

Steganography scenario 2: covert data storage and distribution. Data storage can be viewed as a communication channel between the user and the user himself in the future. Unlike covert communication, covert persistent storage requires the uploaded content to be permanently stored and available on the blockchain. Given the cost of securely transmitting 2KB in Bytecoin is $0.0000215, the cost of covertly storing 1MB is about $0.011. Consequently, an adversary can use Bytecoin as a *cyberlocker* and abuse the P2P network of Bytecoin as a persistent content-distribution network (CDN). For example, a blockchain could be used to store pirated movies, WikiLeaks documents, etc.

A hypothetical example that shows the threat of this scenario is *blackmailing*. An adversary can covertly store some private information about a victim, and may even demonstrate this to the victim by sharing the key and the extraction tool with him. The malicious user can then threaten the victim that she can make the information public by revealing the key to everyone. The victim will face difficult options due to the absence of any central authority, which may be able to remove the content, and the permanent storage of transactions in public blockchains.

4.3 Covert Broadcast Communication in Blockchains

In this section, we describe an example implementation of scenario 1, i.e. implementing covert broadcast communication on public blockchains. We modify the kleptographic attack on CryptoNote's ring signature presented in Sec. 3.3, and in particular the attack realization on Bytecoin as described in Sec. 3.6.1, to demonstrate our implementation of the world-first covert broadcast communication application on blockchains. For brevity, we henceforth refer to our application as *Skywhisper*.

[3] Price as shown on `https://coinmarketcap.com/` on 26/June/2020.

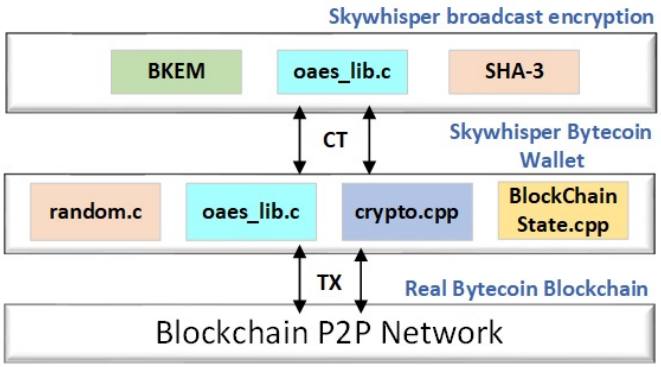


Figure 4.4: Skywhisper layers.

4.3.1 System requirement

As demonstrated in Chapter 3, all public blockchains with randomized signatures are susceptible to various kleptographic attacks. Also, since undetectable kleptographic attacks correspond to secure steganography channels [BL17], in principle, all public blockchains with randomized signatures can be abused to establish covert broadcast communication. However, the available bandwidth per transaction depends on the specific signature scheme. In our example implementation, we take advantage of the random numbers within the ring signature(s) of each transaction in Bytecoin. The available bandwidth per transaction allows sending key-management commands in a single transaction. The size of the biggest command in our particular implementation is 1237 bytes, that is used to manage a broadcast channel of 64 users, as described later in Table 4.1. Public blockchains with randomized signatures that offer less bandwidth can send these commands over multiple transactions, which is less efficient as compared to Bytecoin and, generally, cryptocurrencies that use ring signatures. Therefore, all public blockchains with randomized primitives can be exploited to realize steganographic systems; however, their actual implementation depends on the used cryptographic primitives.

4.3.2 System architecture

Skywhisper is a covert broadcast communication system that can, in theory, be deployed on any existing blockchain platform with randomized cryptographic primitives. The prototype is developed and tested on the real-world Bytecoin blockchain (v3.3.3). Its

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

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

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

Table 4.1: Structure of Skywhisper broadcast packets.

general objective is to override Bytecoin and use it for secure covert communication and persistent storage. As illustrated in Fig. 4.4, Skywhisper consists of three layers: (1) Skywhisper broadcast encryption, (2) Skywhisper Bytecoin wallet, and (3) Bytecoin P2P network. The third layer is responsible for providing *anonymous* connectivity among the users and comprises the normal Bytecoin P2P network. Notably, to run a Skywhisper node, users do not require any additional infrastructure or hardware as long as they have a Skywhisper-modified Bytecoin wallet. The following two sections describe the first two layers of Skywhisper.

4.3.3 Skywhisper broadcast encryption

The core building block of this layer is *broadcast encryption* pioneered by Fiat and Naor [FN94]. Broadcast encryption is a type of encryption where one ciphertext is transmitted in a broadcast channel to all n users in a group $\mathcal{U}$. However, only privileged

users $\mathcal{S}$ can correctly decrypt it using their respective private keys. Where $\mathcal{S}$ is any subset of $\mathcal{U}$, i.e. $\mathcal{S} \subseteq \mathcal{U}$. More specifically, we use Boneh et al.'s [BGW05] broadcast encryption scheme $\mathcal{BE}$, which is a tuple of three algorithms: $\mathcal{BE} = (\mathsf{Setup}, \mathsf{Enc}, \mathsf{Dec})$, as follows:

- $(\{d_1, \ldots, d_n\}, \mathsf{PK}) \leftarrow \mathsf{Setup}(n)$. The setup algorithm takes the number of users n as input, and generates one public key PK and a set of n private keys $\{d_1, \ldots, d_n\}$.
- $(\mathsf{Hdr}, \mathsf{K}) \leftarrow \mathsf{Enc}(\mathcal{S}, \mathsf{PK})$. The encryption algorithm takes as input a subset of privileged users $\mathcal{S} \subseteq \mathcal{U}$ and a public key PK. It is run by the master node to generate a new header Hdr and a symmetric encryption key K. Hdr is shared with subscriber nodes so that they can generate the symmetric key K, which is used, with a secure encryption algorithm, to encrypt and decrypt any broadcast message.
- $\mathsf{K} \leftarrow \mathsf{Dec}(\mathsf{Hdr}, \mathcal{S}, i, d_i, \mathsf{PK})$. $\forall\ i \in \mathcal{S}$, any subscriber with index i and private key d_i can compute the symmetric broadcast encryption key K.

$\mathcal{BE}$ satisfies *correctness* if the following is true: $\forall\ \mathcal{S} \subseteq \mathcal{U}$ and $\forall\ i \in \mathcal{S}$, if $(\{d_1, .., d_n\}, \mathsf{PK}) \leftarrow \mathsf{Setup}(n)$ and $(\mathsf{Hdr}, \mathsf{K}) \leftarrow \mathsf{Enc}(\mathcal{S}, \mathsf{PK})$, then $\mathsf{K} \leftarrow \mathsf{Dec}(\mathsf{Hdr}, \mathcal{S}, i, d_i, \mathsf{PK})$. Moreover, $\mathcal{BE}$ is said to be *collusion-resistant* if all non-privileged users in $\mathcal{T}$, where $\mathcal{T} \subseteq \mathcal{U}$ and $\mathcal{T} \nsubseteq \mathcal{S}$, can not recover the broadcast secrets even if they collude with each other. The *semantic security* of $\mathcal{BE}$ is formally proved by Boneh et al. [BGW05].

As seen in Fig. 4.4, the *Skywhisper broadcast encryption* layer mainly consists of the following three algorithms. (1) *Broadcast Key-Encapsulation Mechanism (BKEM)*, which is a modified version of an open-source C-library [Gün12] that implements Boneh et al.'s broadcast encryption protocol [BGW05]. According to this protocol, let n be the total number of users in a broadcast channel, then the size of the public key PK and the header Hdr is $O(\sqrt{n})$, and the size of the private keys d_i's is just a single group element. Whenever the subscribers' list is modified, the master node generates a new Hdr and derives a new symmetric encryption key K from the updated Hdr. (2) *AES encryption*, which is included in the Bytecoin wallet source code. It is used to encrypt broadcast messages under the symmetric broadcast key K in the CBC mode. (3) *SHA-3* is used to hash the different Skywhisper commands to ensure authenticity and integrity.

The output of the broadcast encryption component is a broadcast packet, denoted

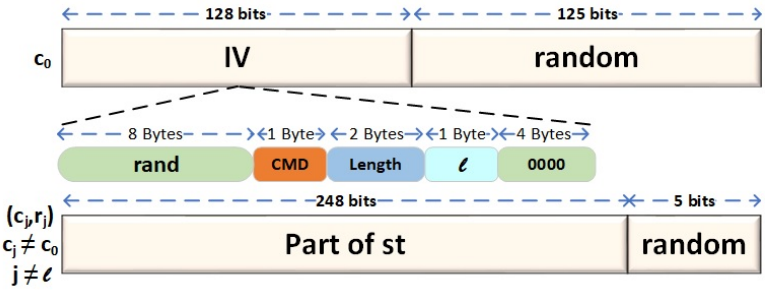


Figure 4.5: Embedding a broadcast packet CT. First generate a synthetic IV = $\text{AES}_z(\text{rand}\|\text{CMD}\|\text{Length}\|\ell\|0000)$ and embed it in c_0. Then generate a stegotext $\text{st} = \text{AES}_z(\text{CT})$, and embed up to 31 bytes of st in each random number c_j and r_j.

as CT, that is either one of the following three commands: (i) subscription-request, (ii) new-Hdr, and (iii) broadcast-msg. The detailed syntax of the three types of CT packets is shown in Table 4.1. After executing Skywhisper's broadcast encryption, the generated CT is passed to the second layer, i.e. the *Skywhisper Bytecoin wallet*, which secretly embeds CT into innocuous-looking transactions.

4.3.4 Skywhisper wallet

This layer consists of a modified version of the Bytecoin wallet (v3.3.3)[4]. Bytecoin is an open-source cryptocurrency project [Tea18], and it uses the ED25519 twisted Edwards curve and the CryptoNote linkable ring signature to sign its transactions. As shown in Sec. 2.2.6.2, this signature has sufficiently many uncontrolled random numbers. For a ring of size k, i.e. k public keys, the format of the ring signature is $\sigma = (I, c_1, \ldots, c_k, r_1, \ldots, r_k)$. For all $j \in [k]$, where $j \neq \ell$ and ℓ is the signer's index, the components c_j and r_j are uncontrolled random group elements in $\mathbb{Z}_p$. These elements can be abused to establish subliminal channels into which the broadcast communication is secretly embedded. The Skywhisper wallet achieves steganographic communication according to the following three steps.

Step 1: embedding a broadcast packet CT in a transaction's signature. To covertly embed CT in signature's random elements $\{(c_j, r_j)\}$, the modified wallet generates a synthetic $\text{IV} = \text{AES}_z(\text{rand}\|\text{CMD}\|\text{Length}\|\ell\|0000)$. Here z is a 128-bit channel key shared

[4] Although slightly different, we have confirmed the same principles are still applicable, and Skywhisper can be implemented with subtle modifications in Bytecoin's latest release (v3.5.1).

among all n users, rand is a 64-bit random string, CMD signifies one of the three packets in Table 4.1, Length is the length of CT, ℓ is the signer's secret index within the ring signature, and 0000 is a 32-bit string of 0's. Then IV is placed in the least significant 16 bytes of c_0, and the rest of c_0 is set randomly. After that, using AES-128-CBC, z, and IV, the broadcaster generates a steganographic text st by encrypting CT; i.e. $\mathsf{st} = \mathrm{AES}_z(\mathsf{CT})$. Then, as illustrated in Fig. 4.5, the broadcaster places parts of up to 31 bytes of st in all subsequent random numbers until the end of st. Finally, the transaction that contains st is sent as per normal over the blockchain.

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

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

The changes introduced to the normal Bytecoin wallet affect three source files: crypto.cpp, BlockchainState.cpp, and oaes_lib.c. For further clarity, the pseudo-code of the Skywhisper Bytecoin wallet is shown in Fig. 4.6. Also, the source code for Skywhisper is available in GitHub[5] along with instructions on how to compile it and execute Skywhisper's different operations.

[5] `https://github.com/NaLancaster/skywhisper`

Bytecoin stegosystem pseudo-code

$\mathsf{KeyGen}(1^{128})$**:**

- Pick random $z \xleftarrow{\$} \{0,1\}^{128}$;
- Return $\mathsf{ek} = \mathsf{dk} = z$;

$\mathsf{Embed}_z(\mathsf{CT})$:

generate_ring_signature()**:**

- counter $= 0$;
- if(j $\neq \ell$):
  - $c_j = \mathsf{random\_scalar}(\ell, \mathsf{counter}{+}{+})$;
  - $r_j = \mathsf{random\_scalar}(\ell, \mathsf{counter}{+}{+})$;
- Else:
  - process as per normal;

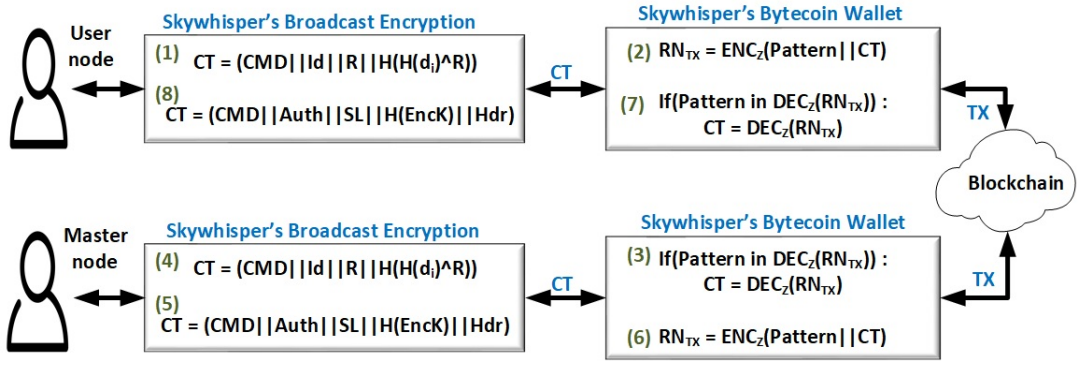
$\mathsf{random\_scalar}(\ell, \mathsf{counter})$**:**

- rand $\xleftarrow{\$} \mathbb{Z}_p$;
- if(counter == 0):
  - $\mathsf{IV} = \mathsf{Encrypt}_z(\mathsf{rand}_{[0:63]}||\mathsf{CMD}||\mathsf{Length}||\ell||\mathsf{zeros})$;
  - $\mathsf{st} = \mathsf{Encrypt}_{z,\mathsf{IV}}(\mathsf{CT})$;
  - $\mathsf{rand}_{[0:127]} = \mathsf{IV}$;
  - sent $= 0$;
- if(counter > 0):
  - if(sent $<$ Length):
    - $\mathsf{rand}_{[0:247]}$ = up to 31 bytes of st;
    - sent = sent + (up to 31);
- Return rand;

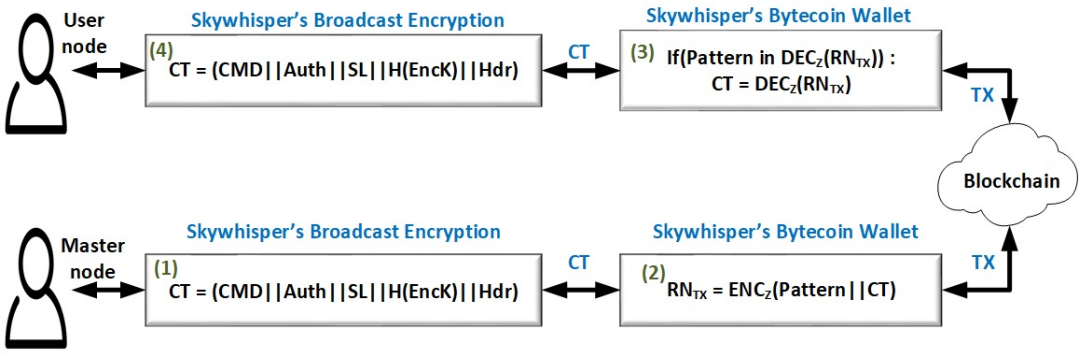
$\mathsf{Extract}_z(c, r)$:

- pattern_found $= 0$;
- for$(j = 0; j < 2; j{+}{+})$
  - $\mathsf{IV}' = \mathsf{Encrypt}_z^{-1}(c_{j,[0:127]})$;
  - if$(\mathsf{IV}'_{[96:127]} == \mathsf{zeros})$:
    - $\mathsf{CMD} = \mathsf{IV}'_{[64:71]}$ $\mathsf{Length} = \mathsf{IV}'_{[72:87]}$ $\ell = \mathsf{IV}'_{[88:95]}$;
    - pattern $= 1$, index $= j$;
- if(pattern):
  - for$(j = 0; j < k; j{+}{+})$
    if$(j \neq \ell \;\&\; j \neq \mathsf{index})$: $\mathsf{CT} = \mathsf{CT} + \mathsf{Encrypt}_{z,\mathsf{IV}}^{-1}(c_j, r_j)$;
  - Return CT;
- Return 0; % *No broadcast message*

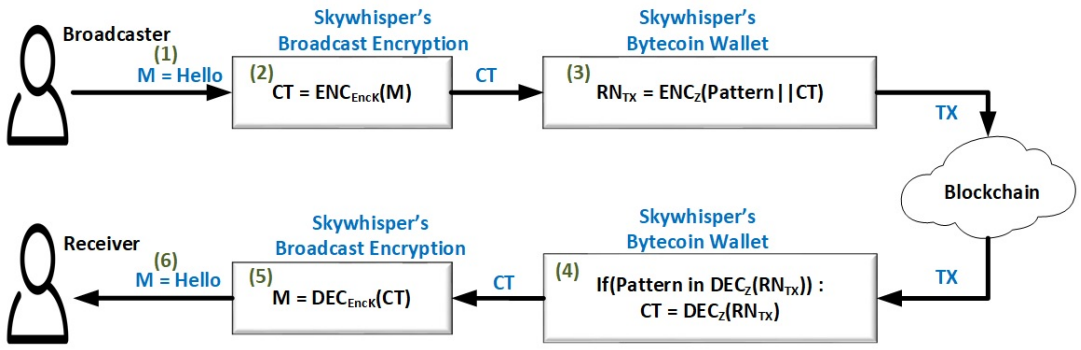
Figure 4.6: Pseudo-code for the implementation of the stegosystem $\mathcal{ST}$ in the Skywhisper's modified Bytecoin wallet. k denotes the size of the ring signature, ℓ is the signer's index within the ring, and z is the steganographic channel key.



(a) User subscription



(b) User revocation



(c) Broadcast communication

Figure 4.7: Skywhisper operations: user subscription and revocation, and broadcast communication. $\mathsf{RN_{TX}}$ denotes the random elements in the ring signature.

4.3.5 System operation

To fully achieve its objectives, Skywhipser offers the following three functionalities: **(1)** user subscription, **(2)** user revocation, and **(3)** broadcast communication. In the following, we describe each of these three operations.

User subscription. As illustrated in Fig. 4.7a, when a user requests a subscription to Skywhisper's broadcast channel, the following steps occur. (1) A broadcast packet CT is created containing 1 byte indicating the subscription-request command, the user's Id, a 32-byte random string R, and 2 LSB's of $\mathsf{Hash}(\mathsf{Hash}(d_i) \oplus \mathsf{R})$ where d_i is the user's private key. (2) After that, the Skywhisper's Bytecoin wallet covertly embeds CT into a transaction along with a known pattern and broadcasts it through the blockchain. (3) When the master node's wallet detects the known pattern, it extracts the broadcast packet CT. (4) Then, the master node's broadcast encryption layer authenticates the subscription request by xoring the received R with the user's secret key d_i and checking the result hash value with the received hash. If successfully authenticated, the user is added to the subscribers' list, and a new Hdr is generated[6]. (5) The master node generates a broadcast packet containing 1 byte indicating the new-Hdr command, some authentication data Auth, the new subscribers' list SL, 12 bytes of the hash of the new encryption key K, and the new Hdr. (6) CT is secretly embedded into a transaction and transmitted through the blockchain. (7) The user node's wallet detects a transaction containing a broadcast packet and extracts CT. (8) After extracting CT, the user checks its data, and if it is successfully authenticated, the user generates a new K using the received Hdr. Finally, the user verifies the correctness of K by comparing its hash to the received hash value.

User revocation. Fig. 4.7b shows the process of revoking the subscription of a given user. The master node initiates this process by removing the user's index from the

[6] *Remark:* the size of Hdr is proportional to the number of users n. Hdr contains $A+1$ elements, where $n = AB$. It is stated in [BGW05] that setting $B = \lfloor \sqrt{n} \rfloor$ gives both public key PK and Hdr of size of about $\sqrt{n}$ elements. Also, the algorithm will set up $A = \lceil \frac{n}{B} \rceil$. In our implementation of Skywhisper, we have $n = 64$, $A = 8$, and $B = 8$, which gives Hdr of 9 elements. Since each element is 128 Bytes, the size of Hdr is 1152 Bytes. After updating the subscribers' list, Skywhisper does not only broadcast Hdr to users but sends a broadcast packet $\mathsf{CT} = (\mathsf{CMD} \| \mathsf{Auth} \| \mathsf{SL} \| \mathsf{H}(\mathsf{K}) \| \mathsf{Hdr})$, which is 1237 bytes as further illustrated in Table 4.1

Throughput and Cost					
Tool	**BW (byte/Tx)**	**Tx Fee (coin)**	**Price/coin**	**Tx Fee ($)**	**Cost 1 MB($)**
Skywhisper	2 K	0.1 BCN	0.000215	0.0000215	≈ 0.011
Tithonus	1650	10^{-8} BTC/byte	9305.01	0.153532665	≈ 97.6
R3C3	1168	0.0001 ZEC	56.09	0.005609	≈ 5

Table 4.2: Comparison between Skywhisper, Tithonus [RC19], and R3C3 [MMSK18] in terms of bandwidth per transaction (BW) and cost of transmitting 1MB, assuming a Bytecoin transaction with 4 inputs and ring size 10.

subscribers' list and generating a new Hdr. After that, the same process is executed as steps (5)-(7) of the user subscription scenario. However, in this case, the revoked user receives a hash value for K that is different from the hash value he computes; hence, the user concludes that his subscription has been revoked.

Broadcast communication. As illustrated in Fig. 4.7c, any subscriber, including the master node, can send a broadcast message M to all other subscribers according to the following steps. (1) The broadcaster constructs a message M. (2) The message M is encrypted under the broadcast encryption key K, and a broadcast packet CT is generated. (3) The sender's Skywhisper wallet encrypts CT under the channel key z, embeds it into the transaction's signatures' random numbers $\mathsf{RN_{TX}}$, and broadcasts the crafted transaction as per usual through the blockchain P2P network. (4) Each receiver's client detects a new transaction and checks if its randomness contains the known pattern. If the pattern is found, using the channel key z, the client extracts the received broadcast ciphertext CT and passes it to the broadcast encryption layer. (5) After that, CT is decrypted using the broadcast encryption key K. (6) Finally, the receiver correctly recovers the broadcast message M.

4.3.6 Performance and security of Skywhisper

Performance. The available bandwidth BW in a Bytecoin transaction, and equally in a CryptoNote transaction, is given as $\mathsf{BW} = 32(y(k-1)2)$ bytes, where y is the number of inputs in the transaction, and k is the number of public keys in each ring signature. Hence, as detailed in Table 4.2, in one transaction of 4 inputs and 10 public keys, Skywhisper can transmit more than 2KB of covert data, which, given the current price of Bytecoin is \$0.000215 [Coi18], costs \$0.0000215. Therefore, the cost of transmitting

1MB of data via Skywhisper is approximately \$0.011. This is much lower compared to other blockchains-based applications such as Tithonous [RC19] and R3C3 [MMSK18], where transmitting 1MB through Tithonous costs $\approx$ \$97.6, and $\approx$ \$5 in R3C3[7]. Note that neither Tithonous nor R3C3 is a broadcast communication tool, and they are rather censorship-circumvention tools.

Security and robustness. To prove that Skywhisper is a secure stegosystem, it should be proven that there is not any PPT observer that can distinguish between its stegotext and innocent cover text better than random, i.e. can win the security game in Fig. 4.1. However, as stated by Katzenbeisser and Petitcolas [KP02], it is difficult to prove this property. They argue that one can, instead, base the security proof of the steganographic system $\mathcal{ST}$ on another problem $\mathcal{P}$ that is known to be secure, e.g. some intractable cryptographic primitive, and construct a reduction from $\mathcal{P}$ to $\mathcal{ST}$.

Similarly, the security of Skywhisper depends on the semantic security of the broadcast encryption scheme $\mathcal{BE}$, as defined in Sec. 4.3.3, and the indistinguishability of the stegosystem $\mathcal{ST}$ implemented in the modified Bytecoin wallet to covertly embed broadcast packets CT in transactions' ring signatures. The proof of the former one can be found in [BGW05]. Whereas, the security of the proposed stegosystem $\mathcal{ST}$ is examined for undetectability under the chosen hidden-text attack (CHA) security game depicted in Fig. 4.1. We remark that other content-insertion techniques that use non-standard Bitcoin scripts or exchange the public key with an arbitrary string containing *printable* characters can be detected [MHH$^+$16, MHH$^+$18]. However, Skywhisper's $\mathcal{ST}$ replaces random group elements in the ring signatures with pseudo-random ciphers, which, by definition of semantic security, are computationally indistinguishable from each other. More formally, we model the underlying encryption scheme $\mathsf{Encrypt}$ of Fig. 4.6 as a pseudo-random function (PRF), and we have the following theorem.

Theorem 4.1. *If* $\mathsf{Encrypt}$ *is a secure pseudo-random function, then the stegosystem* $\mathcal{ST} = (\mathsf{KeyGen}, \mathsf{Embed}, \mathsf{Extract})$*, as shown in Fig. 4.6, is CHA secure.*

[7] Comparison is based on the prices of the related cryptocurrencies quoted on 26/06/2020 from [Coi18].

Proof. In Appendix C (Sec. C.2), we use reduction to prove Theorem 4.1. We show that if there is a PPT adversary $\mathcal{A}$ that breaks the security of Skywhisper with respect to the CHA security experiment $\mathbf{Expt}^{\mathsf{CHA}}_{\mathcal{A}}(1^{\lambda})$ in Fig. 4.1, then there must exist another PPT adversary $\mathcal{B}$ who can break the PRF game for $\mathsf{Encrypt}$, i.e. $\mathcal{B}$ must be able to distinguish, with a non-negligible probability, between the pseudo-random text generated by the PRF and randomly-selected text, as explained in Sec. 2.2.3. □

By contrast, Frkat et al. [FAZ18] showed how to insert arbitrary content in Bitcoin's transactions by replacing the ephemeral randomness r in each transaction's ECDSA signature with the ciphertext of the hidden message. The authors demonstrated their technique in the context of botnets, where a central bot, or botmaster, communicates commands subliminally to other bots in the botnet. Nonetheless, their model suffers a severe security shortcoming represented in the inability of the botmaster to generate the same command twice; otherwise, the botmaster's private key can be computed by any observer. Besides, a warden can detect this steganographic communication when the botmaster communicates the same message more than once. In other words, their scheme is not secure against chosen hidden-text attack, where the warden can detect the steganographic communication by repeating a message twice.

In terms of robustness, as defined in Sec. 4.1, unlike image steganography, the stegotext embedded in the signatures can not be removed without nullifying the functionality of the signatures and the transactions in general. Therefore, Skywhisper is robust against any warden who attempts to alter the stegotext. Remarkably, other content-insertion approaches that replace segments of the transactions, as done in Tithonus [RC19] and Catena [TD17], are susceptible to policy changes where certain transactions, or scripts, become conspicuous or are no longer accepted, forcing the adoption of alternative techniques.

Advantages and disadvantages of Skywhisper. Skywhisper can be used as a censorship-circumvention tool to avoid the shortcomings of other proxies. Particularly, state-of-the-art censors can discover censorship-resistant proxies, e.g. Tor bridges, and block them. On the contrary, it is provably secure that no censor can distinguish

steganographically-subverted blockchain transactions. Hence, censors can not launch any targeted DoS attack against Skywhisper unless they blacklist the whole blockchain network, which might have other financial ramifications.

Besides its security and robustness, Skywhisper offers the following advantages. By utilizing public blockchains, *Skywhisper eliminates the need for dedicated private blockchains for lightweight communication applications.* It makes use of well-established public blockchains, which are backed by many nodes that participate in the consensus process incentivized by the already-available cryptocurrency. Also, *Skywhisper delegates computation to outside the blockchain.* This delegation removes the need for a Turing-complete programming language for the blockchain platform, uses well-tested and widely-known programming languages, and speeds up processing. Besides, *Skywhisper offers interoperability* where Skywhisper master and subscribers transact seamlessly with other clients in the blockchain.

On the other hand, the main disadvantage of Skywhisper is the fact that it can be used by outlaws to abuse public blockchains for illicit communication and storage. Specifically, they can exploit steganographic communication in blockchains to circumvent legal censorship, and secretly store and disseminate objectionable content. This potential threat necessitates the research of countermeasures to deter the steganographic exploitation of blockchains, as demonstrated by Skywhisper, and prevent kleptographic attacks on cryptocurrencies as previously described in Chapter 3.

4.4 Summary of Steganography in Blockchains

Ateniese et al. [BL17] have shown that kleptographic attacks correspond to secure steganography on specific randomized cryptographic schemes. The work presented in this chapter demonstrates this correspondence on randomized signature schemes in public blockchains. In this chapter, we have shown that our kleptographic attack on Bytecoin's ring signature, described in Chapter 3, can be used to realize steganographic communication in Bytecoin. We have demonstrated how to use this steganographic communication to implement the first blockchain-based covert broadcast communication

tool, called *Skywhisper*. This tool presents a robust provably-secure stegosystem, with a broadcast network of 64 users, and can transmit 1MB of data for about $0.011.

Although steganography in public blockchains can help realize innovative benign solutions on top of existing blockchains, such as Skywhisper, we have clarified that steganography has severe adversarial effects. Namely, steganographic communication in public blockchains enables the covert storage and dissemination of objectionable content. Such abuse of public blockchains will motivate regulators to criminalize the use of blockchains that are known to store unethical content. Besides, users will inevitably entirely abandon the use of blockchains or opt-out of the participation in the consensus protocol. Consequently, steganography in blockchains hinders the future adoption of the technology. Therefore, new blockchain designs should actively prevent steganographic abuse of public blockchains.

Chapter 5

Existing Countermeasures

Previous chapters have highlighted the potential threat of exploiting the randomness in public blockchains' cryptographic primitives. In particular, we demonstrated the threat of kleptographic and steganographic attacks on blockchains in Chapters 3 and 4, respectively. Besides, we illustrated how these attacks could be used to maliciously subvert blockchain applications, communicate covertly, and irrevocably store arbitrary data in blockchains. Besides, we clarified the adverse repercussions on the unknowing users of kleptographically-subverted wallets, who are inevitably vulnerable to losing their secret signing keys and, consequently, their funds. We have also shed light on the regulatory challenge posed by storing arbitrary, and possibly unethical, content in blockchains. Also, we discussed how storing malicious content might lead to the legal criminalization of the mere participation in blockchains, which are known to contain such content.

Given the aforementioned threats, it is necessary to investigate possible countermeasures to deter the kleptographic subversion of blockchain applications, prevent steganographic communication, and actively remove arbitrary content before it is added to the ledger. The latter objective of any efficient countermeasure does not only defend against the permanent storage of malicious data but also *practically* achieves the 'Right to be Forgotten' (RtbF), which is required by regulators, such as the European General Data Protection Regulation (GDPR) [PCAP19].

To investigate possible countermeasures, we survey in this chapter all the state-of-the-art techniques that are proposed as deterrence against kleptography and steganography in general, and particularly in blockchains. We assess the efficiency of each of the surveyed techniques using the following five metrics. (1) The ability of the technique to thwart the kleptographic subversion of blockchain applications. (2) The ability of the technique to defend against steganographic communication. (3) The efficiency of the technique in deterring the persistent storage of arbitrary content. (4) The resistance of the technique against input-triggered attacks. (5) The technique's practical applicability in the context of blockchains. We denote by *input-triggered* attacks the scenarios where the maliciously-implemented cryptographic algorithm behaves honestly, i.e. as per specifications, except when triggered by a specific input chosen by the attacker. Also, we use *practical applicability* to indicate the feasibility of practically implementing the described countermeasure in the context of public blockchains.

Sec. 5.1 lists all of the previously proposed countermeasures, describes each one of them, and assesses their effectiveness with respect to the above-mentioned metrics. After that, Sec. 5.2 discusses some of the new blockchain design trends and assesses them in light of their effectiveness as countermeasures. Finally, Sec. 5.3 presents an overall summary of all the surveyed techniques.

5.1 Kleptography-Resistant Techniques

5.1.1 De-randomization of algorithms

As demonstrated in Chapters 3 and 4, kleptographic and steganographic attacks on cryptographic primitives exploit the uncontrolled randomness within these primitives. Thus, the first intuitive countermeasure, as proposed by Young and Yung [YY96], is to de-randomize the random algorithm by splitting it into two parts: a deterministic algorithm and an external random-number-generator algorithm. This approach allows the users to choose their random parameters, e.g. seeds, and make the random algorithms

publicly available. Consequently, users can compare the output of their de-randomized algorithms with the output of the corresponding trusted implementation, given the same input parameters are used in both.

Although this approach is theoretically possible, it fails to address *input-triggered* kleptographic attacks. A user-side implementation can act honestly for most inputs except when the input is equal to some trigger value that is specified by the implementer. Besides, the de-randomization of algorithms is not effective against steganography, where the user willingly chooses to bypass the randomness generation algorithm and knowingly replaces it with pseudo-random-looking stegotext.

5.1.2 Cascading cryptosystems

Another measure proposed by Young and Yung to counter kleptography is to cascade independently-developed cryptosystems [YY96]. The use of multiple cryptosystems avoids the unreasonable trust of a single cryptosystem in the black-box setting and thwarts kleptography in principle. However, it is not practical in the context of blockchain applications. Also, this approach will likely increase the computational time, and users will need to use multiple applications or hardware devices to establish the needed level of trust. Like the previous technique in Sec. 5.1.1, this technique is not effective against steganography.

5.1.3 Software integrity

Young and Yung recommended checking the integrity of cryptographic software as a countermeasure against kleptography [YY96]. To achieve this, deterministic compilation, also known as reproducible builds [rep20], and tools such as Gitian [Cup20], can be used to ensure the same source code compiles into the same binary even if compiled in different environments. Kleptography can be prevented if the used software successfully passes sufficient integrity checks, or if the binary is deterministically compiled from trusted source code. However, deterministic compilation can not deter steganography in blockchains, as users knowingly choose to use steganography-enabled software.

5.1.4 Synthetically-random cryptographic primitives

As identified by many researchers [YY96, YY97a, BPR14, BJK15], the root cause for kleptographic attacks is the *uncontrolled randomness*. As such, an intuitive solution is to use deterministic cryptographic primitives [BPR14, BJK15, BH15, DFP15]. Concerning signature schemes, one can replace the randomized ECDSA signature scheme, for example, by the synthetically-random EdDSA signature [BDL+11]. In fact, the ECDSA signature can be changed to use synthetic randomness as proposed by RFC 6979 [Por13]. Likewise, Ateniese et al. [AMV15] have proved that signature schemes with unique signatures are subversion-resilient against attacks that meet a basic undetectability requirement.

Indeed, if implemented according to specifications, the use of synthetically-random primitives effectively thwarts kleptography and steganography abuse; however, synthetic randomness can be exploited using hidden triggers. Namely, assume a signature algorithm consumes n random coins, denoted as $r_1, \ldots, r_n$. Without loss of generality, suppose the signing algorithm takes as input the signing key s and the message m. We can generate the needed random coins synthetically as $r_i = \mathsf{hash}(s, m, i)$. Based on the heuristics property and onewayness of the hash function, r_i is unpredictable due to the entropy of s. On the other hand, this tweak allows offline watchdogs, i.e. verification algorithms, to compare an implementation with its specification.

Nevertheless, no probabilistic polynomial-time black-box verification mechanism can ensure an implementation exactly matches its specification. This is because a malicious functionality may be triggered by a specific input, and it is computationally infeasible to verify that an implementation behaves as expected for all inputs. For instance, a synthetically-random signing algorithm can be implemented so that it behaves honestly for all the inputs, except when the input message $m = m^*$ the signing algorithm switches to its malicious behaviour, where m^* is a hidden trigger that has a high entropy. Hence, the use of synthetically-random signature schemes is not a practical countermeasure against kleptography because of possible hidden triggers. Also, such signatures can not defend against steganography since the synthetic randomness can be bypassed without it being detected by any warden who is scrutinizing the generated signatures.

5.1.5 VRFs and controlled randomness

Verifiable Random Functions (VRF), introduced by Micali et al. [MRV99], are random functions that non-interactively prove the correctness of their outputs. A function F is a VRF if there exists a tuple of three algorithms $(\mathsf{Generate}, \mathsf{Prove}, \mathsf{Verify})$ with the following characteristics. On the input of certain security parameter(s), $\mathsf{Generate}$ outputs a pair of keys: a secret key sk and a public key PK. On input of sk and a value x, Prove outputs (y, π), where $y = \mathsf{F}_{sk}(x)$ and π is the corresponding proof of correctness, such that $\mathsf{Verify}_{PK}(x, y, \pi)$ verifies successfully *only* if $y = \mathsf{F}_{sk}(x)$ [DY05]. A similar approach was suggested by Bohli et al. [BGVS07], who proposed a subliminal-free variant of ECDSA that requires non-interactive wardens.

Similarly, Hanzlik et al. [HKK17] presented a deterministic approach to verify the output of a pseudo-random number generator (PRNG) that is installed as a black-box by the manufacturer. In their approach, the user provides the PRNG with a blinding factor $U = g^u$, where u is kept secret by the user. Each time the PRNG generates a random number r, it also outputs a witness $(\acute{r}, i)$, which can be used with u to verify the honest generation of r.

Both VRFs [MRV99] and the controlled randomness approach [HKK17] have two obvious shortcomings when considered as countermeasures against kleptography and steganography in cryptographic primitives. The first shortcoming is their susceptibility to rejection sampling. For example, in VRFs, the output y can be biased by trying different x values. Besides, some primitives, such as ring signatures [RST01], contain some elements that are generated randomly, and other elements that are generated pseudo-randomly, and both types of elements should not be distinguishable by any probabilistic polynomial-time algorithm, which is not attainable when using VRFs or the controlled randomness approach. In addition, the controlled randomness approach by Hanzlik et al. [HKK17] is specifically useful in combating kleptography. However, a user can bypass this approach entirely and generate random numbers with hidden steganographic content. Hence, although these two approaches may decrease the available bandwidth for kleptography and steganography, and increase their computational cost;

still, due to the previous shortcomings, they are not practical to thwart kleptography and steganography on cryptographic primitives.

5.1.6 Split-program model

The split-program model, proposed by Russell et al. [RTYZ16a], attempts to make a general randomized algorithm G resistant to kleptography by breaking G into two algorithms $(\mathsf{RG}, \mathsf{dG})$, and adding a deterministic immunizing algorithm ϕ, such as a public hash function. Russell et al. used this kleptography-resistant randomized algorithm G to establish security in the kleptographic settings for several primitives, such as one-way permutations (OWP), trap-door one-way permutations (TDOWP), and signature schemes based on their kleptography-resistant TDOWP [RTYZ16a]. As argued by Russell et al. [RTYZ16b], the split-program model in [RTYZ16a] is used in the construction of very specific primitives and does not provide protection against arbitrary kleptographic attacks.

To address the shortcomings of the split-program model in [RTYZ16a], this model was further developed in subsequent work [RTYZ16b, RTYZ17] by decomposing each randomness-generation function RG into two separate components $\mathsf{RG_0}$ and $\mathsf{RG_1}$, executing them independently, and amalgamating their output using a public immunizing function ϕ to generate the final random number. Although not in the split-program model, a similar concept of immunizing functions was also proposed by Dodis et al. [DGG+15] to sanitize the output of backdoored pseudo-random generators.

As emphasized in [RTYZ17], the practical implementation of such splitting necessitates the independent execution of the separate components, which can be achieved by executing them in separate environments, e.g. by running them in different virtual machines or containers like Docker [DI18]. Importantly, this technique is not effective against input-triggered attacks and does not immunize randomized schemes against steganography since the user can bypass the split-program model.

5.1.7 Interactive warden

To defeat subliminal-channels in Schnorr signatures, Zhang et al. [ZLLZ13] proposed a subliminal-free variant of the Schnorr signature [Sch91] using an honest-but-curious interactive warden with whom the signer exchanges a total of six messages. The existential unforgeability of their signature is proven under the computational Diffie-Hellman assumption. Also, the difficulty of embedding subliminal channels in the created signatures is shown to be equal to solving the discrete-logarithm problem.

As criticized in [HAZ17], although this scheme can deter kleptographic and steganographic attacks on blockchains, this scheme's main shortcomings are its large number of exchanged messages and the computational cost incurred on the signer and the warden. Besides, this scheme is impractical in public blockchains as it defeats their decentralized design and requires the existence of trusted wardens, which is not attainable in current blockchains.

5.1.8 Reverse firewalls (RF)

The concept of a cryptographic reverse firewall (RF) was introduced by Mironov and Stephens-Davidowitz [MSD15] to immunize cryptographic schemes against kleptographic attacks. An RF is a third party that sits between the possibly-subverted algorithm and the outside world. Suppose Alice is a user who is running a possibly corrupted algorithm. An RF can be used to prevent Alice's algorithm from undermining her security by sanitizing her outgoing transactions. Importantly, an RF must use public information *only*, preserve security, maintain functionality, and prevent the exfiltration of any information. Equally important, Alice's algorithm should not place any trust with regard to security or functionality on the reverse firewall and should work seamlessly regardless of the existence or not of such an RF [MSD15, DMSD16].

Furthermore, Ateniese et al. [AMV15] used an untamperable reverse firewall (RF) with re-randomizable signature schemes to construct kleptography-resistant signatures. In this context, RF sits between the possibly subverted signature scheme and the outside world. The RF takes as input a message and its signature σ, and verifies that σ is valid

using public parameters. If the signature σ is valid, the RF re-randomizes σ, and outputs a new signature $\acute{\sigma}$. Although this technique is effective against kleptography attacks and steganography abuse of randomized signatures in blockchains, it requires an active firewall and works only on re-randomizable signatures. Therefore, it is not suitable for the decentralized censor-less nature of blockchains, and it does not offer a universal immunization for all randomized primitives.

5.1.9 Self-guarding protocols

Recently Fischlin and Mazaheri [FM18] proposed a novel technique that proactively defends against kleptographic attacks assuming an initial *temporary trust*, i.e. subversion happens after a period of an honest initial phase. Using this trusted initial phase, they provided kleptography-resistant constructions for homomorphic public-key encryption, symmetric-key encryption, signature schemes, and physically unclonable function PUF-based key exchange. In general, their constructions are divided into two phases: a sampling phase and a challenge phase. In the sampling phase, or honest initial phase, a sample of ciphers, in the case of encryption, or signatures, in the case of signature schemes, is honestly generated. These samples are stored and used in the second phase to obfuscate ciphers/signatures and detect possible kleptographic attacks.

To further illustrate this technique, Fig. 5.1 presents a simplified pseudo-code for the construction of a self-guarding signature scheme $\mathcal{S}^{sg} = (\mathsf{KeyGen}^{sg}, \mathsf{Sign}^{sg}, \mathsf{Verify}^{sg})$ from an underlying deterministic signature scheme $\mathcal{S} = (\mathsf{KeyGen}, \mathsf{Sign}, \mathsf{Verify})$ [FM18]. As seen in this figure, the key generation algorithm KeyGen^{sg}, given a security parameter (1^{λ}), generates a list of λ key pairs (sk_i, pk_i), and sets the private key $sk^{sg} = (sk_1, \ldots, sk_\lambda)$ and the public key $pk^{sg} = (pk_1, \ldots, pk_\lambda)$. As part of the *trusted* sampling phase, $\mathsf{Sample}(sk^{sg})$ is executed to generate a queue of λ pairs of randomly generated messages $m_{r,i}$'s and their corresponding signatures $\sigma_{r,i}$. When signing a message m with the *possibly subverted* signing algorithm Sign, the self-guarding signing algorithm Sign^{sg} does the following. For each $i = \{1, \ldots, \lambda\}$, Sign^{sg} randomly picks $b_i \xleftarrow{\$} \{0, 1\}$, and executes the original signing algorithm Sign twice, once to sign $m_{r,i}$ and another time to sign $(m_{r,i} \oplus [m||\sigma_{r,i}])$.

$\mathcal{S}^{sg}(\mathsf{KeyGen}^{sg}, \mathsf{Sample}, \mathsf{Sign}^{sg}, \mathsf{Verify}^{sg})$

$\mathsf{KeyGen}^{sg}(1^\lambda)$:

- For $i = 1 \ldots \lambda$: $(sk_i, pk_i) \xleftarrow{\$} \mathcal{S}.\mathsf{KeyGen}(1^\lambda)$;
- $(sk^{sg}, pk^{sg}) = ((sk_1, \ldots, sk_\lambda), (pk_1, \ldots, pk_\lambda))$;
- Return (sk^{sg}, pk^{sg});

$\mathsf{Sample}(sk^{sg})$:

- Empty $\mathcal{Q}$;
- For $i = 1 \ldots \lambda$:
  $m_{r,i} \xleftarrow{\$} \{0,1\}^\ell$;
  $\sigma_{r,i} = \mathsf{Sign}(sk_i, m_{r,i})$; //Sign underlying signature scheme is honest in initial phase.
  Store $(m_{r,i}, \sigma_{r,i})$ in $\mathcal{Q}$;
- Return $\mathcal{Q}$;

$\mathsf{Sign}^{sg}(sk^{sg}, m, \mathcal{Q})$:

- if $\mathcal{Q}$ is empty: Return $\perp$;
- For $i = 1 \ldots \lambda$:
  Retrieve $(m_{r,i}, \sigma_{r,i})$ from $\mathcal{Q}$;
  $b_i \xleftarrow{\$} \{0,1\}$;
  if $(b_i = 0)$: $(m^0, m^1) = (m_{r,i}, m_{r,i} \oplus [m||\sigma_{r,i}])$;
  else: $(m^0, m^1) = (m_{r,i} \oplus [m||\sigma_{r,i}], m_{r,i})$;
  $\sigma^0 = \mathsf{Sign}(sk_i, m^0)$; $\sigma^1 = \mathsf{Sign}(sk_i, m^1)$; //Sign underlying signature scheme, which can be potentially subverted.
  if $(\sigma^{b_i} \neq \sigma_{r,i})$: Return $\perp$;
  $\sigma_i = \sigma^{(1-b_i)}$;
- $\sigma = (m_{r,1}, \sigma_{r,1}, \sigma_1, \ldots, m_{r,\lambda}, \sigma_{r,\lambda}, \sigma_\lambda)$;
- Return σ;

$\mathsf{Verify}^{sg}(pk^{sg}, m, \sigma)$:

- result = true;
- For $i = 1 \ldots \lambda$:
  result = result $\wedge$ $\mathsf{Verify}(pk_i, m_{r,i}, \sigma_{r,i})$;
  result = result $\wedge$ $\mathsf{Verify}(pk_i, (m_{r,i} \oplus [m||\sigma_{r,i}], \sigma_i)$;
- Return result;

Figure 5.1: Self-guarding signature $\mathcal{S}^{sg}$ from an underlying signature scheme $\mathcal{S} = (\mathsf{KeyGen}, \mathsf{Sign}, \mathsf{Verify})$. $\mathcal{Q}$ represents a queue of pairs of random messages $m_{r,i}$ and their honestly-generated signatures $\sigma_{r,i}$, λ is a security parameter, and ℓ is the message and signature space length. Other sanity checks have been omitted for simplicity, for more details refer to [FM18].

The order of which is signed first is determined by the value of b_i, and each time the generated signature for $m_{r,i}$ is compared with its previously and *honestly* generated signature $\sigma_{r,i}$. *This is done to ensure detecting any possible corruption with probability* 1/2. If any discrepancy is detected, Sign^{sg} aborts, otherwise returns the signature

$\sigma = (m_{r,1}, \sigma_{r,1}, \sigma_1, \ldots, m_{r,\lambda}, \sigma_{r,\lambda}, \sigma_\lambda)$. The verification Verify^{sg} is carried out by re-constructing the string $(m_{r,i} \oplus [m||\sigma_{r,i}])$ for each $i = \{1, \ldots, \lambda\}$, and calling the original verification algorithm Verify twice, to verify each σ_i and $\sigma_{r,i}$. If all signatures are valid, Verify^{sg} returns true, and false otherwise.

Fischlin and Mazaheri [FM18] have proved the unforgeability of the self-guarding signature scheme $\mathcal{S}^{sg}$ if the underlying signature scheme $\mathcal{S}$ is a deterministic and unforgeable signature. However, they have not shown the applicability of their construction to randomized signatures. Besides, assuming an initial trusted state is not practical in blockchain applications where a client can be kleptographically-corrupted since the time it is first used by the end-user. Moreover, this technique does not provide any assurance against the steganographic abuse of blockchain applications.

5.2 Current Blockchain-Focused Techniques

In this section, we focus on new trends in the design of blockchains; lightweight, prunable, and redactable, and discuss their susceptibility to kleptography and steganography. Also, Sec. 5.2.4 explains some practices that have been recommended to combat the insertion of arbitrary content in blockchains.

5.2.1 Lightweight blockchains

Lightweight blockchains represent a new design trend that aims to solve issues related to blockchain sustainability and scalability. In general, lightweight blockchains either store parts of the transactions off-chain or drop blocks that are older than a certain point. Lightweight blockchains are essential in countering permanent storage. Politou et al. [PCAP19] suggested that storing data in off-chain storage or encrypted storage bypasses the inherent immutability of blockchains and makes them comply with the right to be forgotten (RtbF) as required by the European General Data Protection Regulation (GDPR) [Eur16].

An example of lightweight blockchains is PascalCoin [MS17], which is a cryptocurrency that does not keep the full history of transactions. PascalCoin maintains the latest 100

blocks in its ledger, and stores the actual account balances in another cryptographic structure called the SafeBox. A very similar approach is used in the mini-blockchain scheme [J.D17], which is implemented by the Cryptonite cryptocurrency [Min18]. The mini-blockchain scheme stores the actual balances in a structure called the *account tree*, which is updated by the transactions in the blockchain. Because new transactions reference the *account tree* and not previous transactions in the blockchain, transactions in older blocks can be discarded. Note, older block headers are still kept in the mini-blockchain.

Lightweight blockchains are mainly proposed to solve scalability issues; however, they can also be used to deter permanent storage of malicious content. Nevertheless, they are not immune against kleptographic attacks, and they are still susceptible to covert steganographic communication.

5.2.2 Prunable blockchains

Florian et al. [FHBS19] presented an approach to enable node operators, i.e. clients, to locally erase the content of given transactions, that they deem unwanted, and avoid storing it locally. According to their approach, if a future transaction references previously erased content, the transaction is deemed invalid unless the future transaction is confirmed and added to a mined block. This approach is promising and does not require changes to the blockchain. Nonetheless, it is presented in a particular context; to erase the content of the transactions' output in UTXO-based blockchains. In addition, this approach assumes the ability of the clients to identify unethical content, which is not feasible in the case of steganography. Even if the identification of such content is possible, and this approach is successful in removing data from persistent storage, this method is not efficient against steganographic communication, nor is it efficient as deterrence against kleptography. Another example of prunable blockchains is the Rollerchain presented by the work of Chepurnoy et al. [CLO16].

5.2.3 Redactable blockchains

Unlike prunable blockchains where data is locally erased, redactable blockchains globally delete previously added content, i.e. mined content, without the need for hard forks [FHBS19]. The Redactable blockchain proposed by Ateniese et al. [AMVA17] uses Chameleon hash [KR00, CDK+17] $\mathcal{C}$, instead of the conventional collision-resistant hash function $\mathcal{H}$, to chain mined blocks. Chameleon hash $\mathcal{C}$ enables a trusted central node, or a group of nodes, which possess $\mathcal{C}$'s secret trapdoor key tk to efficiently find a collision, which allows to rewrite, remove, and insert new blocks in the blockchain. For example, a block whose content is x and hash is (h, δ) can be re-written by a new content $\acute{x}$, then using $\mathcal{C}$ and the secret key tk, a collision can be computed $(h, \acute{\delta})$ so that the hash value stays the same, where h is the hash value and δ is a check string. Similarly, $\mu chain$ [PDC17] proposes a mutable blockchain that is based on consensus where users can issue mutability request transactions.

Redactable blockchains are designed mainly for decentralized services and applications other than cryptocurrencies, also known as Bitcoin 2.0. They may not apply to cryptocurrencies, whose ledger should contain the full history of transactions [AMVA17]. Thus, even if the critical question of who possesses the trapdoor key tk is resolved [PCAP19], for example, through consensus-based voting [DMT19], redactable blockchains are not generic for all blockchain applications. Moreover, redactable blockchains can remove/rewrite malicious content only if it is known to the holder(s) of tk. However, if such content is kept confidential, redactable blockchains will not be effective. Besides, redactable blockchains do not stop malicious data from propagating in the P2P network, and thus do not thwart kleptography nor steganographic communication. Similar shortcomings apply to $\mu chain$ [PDC17] and generally to all blockchains that seek to alter content after being added to mined blocks.

5.2.4 Fees, filters, and self-verifying addresses

Matzutt et al. [MHZ+18] proposed three techniques to foil the insertion of arbitrary content in blockchains: (1) increasing the transaction fee, (2) using content filters, and

(3) using self-verifying addresses. Increasing the transaction fee may not be advisable for promoting blockchains among users, and can unfairly penalize users who rely on large transactions, e.g. exchange services. However, minimum mandatory fees have been proposed as a countermeasure to render content insertion economically infeasible for large transactions [MHZ+18].

Content filters target human-readable strings to detect and reject unwanted content, e.g. rejecting a transaction if its 20-byte destination address has 18 printable characters [MHZ+18]. Nonetheless, these filters are not effective against kleptographically-leaked secrets or steganographic communication.

The goal of self-verifying addresses is to deter content insertion in Bitcoin by using arbitrary addresses. Matzutt et al. [MHZ+18] suggested that instead of sending an address a, c_a is sent in the transaction, where $c_a = (G^a, r, \mathsf{Sign}(G^a||r, a))$, $r = \mathsf{CRC32}(t_1|| \ldots ||t_i)$, and t_i is the transaction corresponding to the i^{th} input. A similar approach is to limit the address space. For example, PascalCoin [MS17] has a finite address space, and accounts are limited but can be associated with any public key. These two approaches can deter the arbitrary manipulation of transactions' addresses. However, it is not a generic countermeasure against kleptographic and steganographic abuse of cryptographic primitives in blockchains.

5.3 Summary

In Sec. 5.1 we listed nine techniques that have been proposed in the literature to counter kleptography or steganography or both. These techniques are: (1) de-randomizing random algorithms, (2) cascading independent cryptosystems, (3) inspecting the integrity of the cryptographic software, (4) deterministic cryptographic primitives, (5) verifiable random functions (VRF), (6) the split-program model, (7) interactive wardens (8) reverse firewalls (RF), and (9) self-guarding protocols. Of these nine techniques, there are three countermeasures, (4), (7), and (8), that can defend against all of the kleptographic and steganographic attack scenarios of Chapters 3 and 4. However, all of these three techniques are impractical in public blockchains for the following reasons. The synthetic

randomness in deterministic or synthetically-random schemes is susceptible to input-triggered attacks. Whereas, the last two techniques defeat the decentralized design of blockchains and require the existence of trusted parties in the form of wardens and firewalls.

After that, we described in Sec. 5.2 three new blockchain design trends: (1) lightweight, (2) prunable, and (3) redactable blockchains. We clarified that although these new designs practically thwart the persistent storage of arbitrary content in blockchains, they are inefficient in countering kleptography and steganographic communication. Finally, we discussed the effect of increasing the transaction fee, using content filters, and using self-verifying addresses. All of these three practices are specific solutions and do not offer generic defence against kleptography and steganography. Finally, due to the shortcomings of all of the techniques surveyed in this chapter, in the next chapter, we propose new efficient countermeasures.

Chapter 6

Proposed Countermeasures

All of the existing countermeasures that are surveyed in the previous chapter are inefficient in countering kleptography and steganography. Hence, in this chapter, we propose four new countermeasures: (1) re-randomizable cryptographic primitives in Sec. 6.1, (2) aggregate signatures in Sec. 6.2, (3) randomizable and aggregatable signatures in Sec. 6.3, and (4) a generic steganography-resistant blockchain framework (SRBF) in Sec. 6.4. All of these countermeasures are presented to immunize digital signatures in public blockchains. However, they can be analogously extended to other cryptographic primitives in blockchains, such as the commonly used zero-knowledge proofs (ZKP). As part of the countermeasures, we present two new cryptographic schemes: a randomizable ring signature (RRS) in Sec. 6.1.1, and a randomizable and aggregatable signature (RAS) in Sec. 6.3. Note that our countermeasures aim to eliminate any steganographic messages hidden inside the cryptographic components, such as the signatures, of a blockchain transaction. Nevertheless, the deterrence of general non-steganographic content-insertion techniques, e.g. inserting arbitrary content in unspendable OP_RETURN Bitcoin transactions, is beyond the scope of this work.

6.1 Re-randomizable Cryptographic Primitives

Without loss of generality, we discuss the use of re-randomizable primitives as a countermeasure against kleptography and steganography by the example of re-randomizable

RandomizableSignature(KeyGen, Sign, Verify, Randomize)

Given three bilinear groups of type 3: $(\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T)$, prime number p, and bilinear mapping: $e(\mathbb{G}_1, \mathbb{G}_2) = \mathbb{G}_T$;

KeyGen:

- Select $\widetilde{g} \xleftarrow{\$} \mathbb{G}_2$, and $(a, b) \xleftarrow{\$} \mathbb{Z}_p^2$;
- Compute $(\widetilde{A}, \widetilde{B}) \leftarrow (\widetilde{g}^a, \widetilde{g}^b)$;
- $PK = (\widetilde{g}, \widetilde{A}, \widetilde{B}) \in \mathbb{G}_2$, and $SK = (a, b) \in \mathbb{Z}_p$;

Sign(m, SK):

- Select $h \xleftarrow{\$} \mathbb{G}_1 \backslash 1_{\mathbb{G}_1}$;
- Return $\sigma \leftarrow (h, h^{(a+bm)})$;

Verify(PK, m, σ):

- Parse σ as (σ_1, σ_2);
- Accept if $\sigma_1 \neq 1_{\mathbb{G}_1}$, and $e(\sigma_1, \widetilde{A} \cdot \widetilde{B}^m) = e(\sigma_2, \widetilde{g})$;

Randomize(σ):

- Parse σ as (σ_1, σ_2);
- Select $r \xleftarrow{\$} \mathbb{Z}_p^*$;
- Return $\acute{\sigma} \leftarrow (\sigma_1^r, \sigma_2^r)$;

Figure 6.1: Pointcheval and Sanders's randomizable signature [PS16].

signatures. The use of re-randomizable signatures coupled with reverse firewalls has been previously proposed by Ateniese et al. [AMV15] to construct kleptography-resistant signatures. However, reverse firewalls are not appropriate for the decentralized nature of public blockchains. Therefore, we propose the use of re-randomizable signatures where P2P nodes, instead of reverse firewalls, are responsible for re-randomizing the signatures. An example of re-randomizable signatures is Pointcheval and Sanders's re-randomizable signature [PS16], which is a short signature that consists of two group elements only, as explained in Fig. 6.1. The resulted signature is re-randomizable, where given $\sigma = (\sigma_1, \sigma_2)$ on m, σ can be re-randomized by randomly selecting $r \xleftarrow{\$} \mathbb{Z}_p^*$, and computing $\acute{\sigma} \leftarrow (\sigma_1^r, \sigma_2^r)$. The new signature $\acute{\sigma}$ is a valid signature on m and can be successfully verified using the original Verify algorithm.

In the context of blockchain applications, re-randomizable cryptographic primitives can deter kleptography and steganography if the P2P nodes in the blockchain network are incentivized to re-randomize the primitives within the transactions before propagating these transactions. However, if nodes are not incentivized, they may choose not to

BGLSRingSignature(KeyGen, RingSign, RingVerify)

Given: g_1 and g_2 are generators of groups $\mathbb{G}_1$ and $\mathbb{G}_2$ respectively. $e : \mathbb{G}_1 \times \mathbb{G}_2 \rightarrow \mathbb{G}_T$ is a bilinear map, and $\psi : \mathbb{G}_1 \rightarrow \mathbb{G}_2$ is a computable isomorphism with $\psi(g_1) = g_2$. $H : \{0,1\}^* \rightarrow \mathbb{G}_2$ is a full-domain hash function.

KeyGen:

- $s \xleftarrow{\$} \mathbb{Z}_p$ and $v = g_1^s$;
- Signer's public and secret keys: $PK_j = v_j \in \mathbb{G}_1$, and $SK_j = s \in \mathbb{Z}_p$;

RingSign($L : \{PK_1, \ldots, PK_n\}, m \in \{0,1\}^*, SK$):

- Select randomly $\forall i \neq j$ and $i \leq n : r_i \leftarrow \mathbb{Z}_p$;
- Compute $h = H(m) \in \mathbb{G}_2$;
- Set $\sigma_j = \left(h/\psi(\prod_{i \neq j} PK_i^{r_i})\right)^{1/s}$;
- Compute $\forall i \neq j$ and $i \leq n : \sigma_i = g_2^{r_i}$;
- Return $\sigma = \langle \sigma_1, \ldots, \sigma_n \rangle$;

RingVerify($L : \{PK_1, \ldots, PK_n\}, m \in \{0,1\}^*, \sigma$):

- Compute $h = H(m)$;
- Accept if $e(g_1, h) = \prod_{i=1}^{n} e(PK_i, \sigma_i)$;

Figure 6.2: Bilinear-pairing-based ring signature [BGLS03].

randomize the transactions, rendering this technique inefficient. Besides, it is important to note that a randomized signature $\acute{\sigma}$ is computationally indistinguishable from its corresponding pre-randomized signature σ. Therefore, there is not a 'computational evidence' that a given signature has ever been re-randomized after it was generated by the signer and before it is added to a mined block.

6.1.1 Randomizable bilinear-pairing-based ring signature (RRS)

We present in this subsection a novel randomizable ring signature scheme RRS. This new primitive signature presents a steganography-resistant substitute to CryptoNote's ring signature [Sab13, RST01]. RRS destroys subliminal channels in ring signatures in the context of public blockchains by allowing the P2P nodes to re-randomize the ring signatures before propagating them to the network.

Based on the BLS short signature [BLS01], Boneh et al. proposed a ring signature scheme [BGLS03], which we henceforth refer to as BGLSRingSignature, as shown in Fig. 6.2. As part of the signing algorithm, the signer generates $(n-1)$ arbitrary random points r_i's, where n is equal to the number of public keys in the ring. These random numbers

RRS(RandomizeRing)

RandomizeRing$(L : \{PK_1, \ldots, PK_n\}, m \in \{0,1\}^*, \sigma)$:

- Select randomly $\forall i \leq n : r_i \xleftarrow{\$} \mathbb{Z}_p$;
- For $(j = 1;\ j \leq n;\ j{+}{+})$:
  - $\mathsf{msk}_j = \psi\left(\prod_{i=1,i<j}^{n} PK_i^{r_i} \cdot \prod_{i=1,i>j}^{n} PK_i^{-r_j}\right)$;
  - $\acute{\sigma_j} = \sigma_j \cdot \mathsf{msk}_j$;
- Return $\acute{\sigma} = \langle \acute{\sigma_1}, \ldots, \acute{\sigma_n} \rangle$;

Figure 6.3: Randomizable ring signature (RRS).

are not included directly into the generated signature σ, and $\sigma_{i \neq j} = g_2^{r_i}$ are included instead. Nonetheless, they are still vulnerable to kleptographic attacks through rejection sampling similar to our rejection-sampling attack on the ECDSA scheme presented in Sec. 3.4.2. Hence, to prevent kleptography on the signature's random numbers r_i's, and remove any possible subliminal channels, in the following, we present a re-randomizable version of the BGLSRingSignature scheme.

Our randomized ring signature RRS is based on BGLSRingSignature [BGLS03], and introduces an additional algorithm, RandomizeRing, which obfuscates all the elements in the ring signature $\langle \sigma_1, \ldots, \sigma_n \rangle$. As shown in Fig. 6.3, each element in the original signature σ_j is masked using $\mathsf{msk}_j = \psi\left(\prod_{i=1,i<j}^{n} PK_i^{r_i} \cdot \prod_{i=1,i>j}^{n} PK_i^{-r_j}\right)$, where r_i's are random scalars in $\mathbb{Z}_p$. The verifier of a randomized RRS signature uses the original verification algorithm of the BGLSRingSignature scheme. The correctness of RRS is premised on two properties. The first is the *bilinearity* property of the BGLSRingSignature scheme, which results in:

$$\prod_{i=1}^{n} e(PK_i, \acute{\sigma_i}) = \prod_{i=1}^{n} e(PK_i, \sigma_i \cdot \mathsf{msk}_i) = \prod_{i=1}^{n} e(PK_i, \sigma_i) \cdot \prod_{i=1}^{n} e(PK_i, \mathsf{msk}_i)$$

The second property, on which the correctness of RRS relies, is the way the mask values are constructed. Namely, msk_i's are constructed so that $\prod_{i=1}^{n} e(PK_i, \mathsf{msk}_i) = 1$. Therefore, $\prod_{i=1}^{n} e(PK_i, \acute{\sigma_i}) = \prod_{i=1}^{n} e(PK_i, \sigma_i)$.

Security of (RRS): The security of RRS is defined by the advantage of an adversary $\mathcal{A}$ in forging a randomized ring signature $\acute{\sigma}$, denoted as $\mathsf{AdvRRS}_{\mathcal{A}}$. Using a security

game that further defines this advantage, Appendix C.3 uses reduction to prove that the security of RRS is equivalent to the security of the underlying BGLSRingSignature scheme.

When used in public blockchains, RRS enables the P2P nodes to re-randomize the signatures before relaying them into the network. This re-randomization obstructs any possible subliminal channels and obfuscates any kleptographically-leaked data. Nevertheless, since a re-randomized signature is computationally indistinguishable from the original signature, there is not any 'computational evidence' that a given signature was ever re-randomized before it is included in a mined block. To overcome this weakness, we present in the following section an aggregate signature that provides such evidence.

BLS(KeyGen, Sign, Verify)

Given three bilinear groups: $(\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T)$, prime number p, and bilinear mapping: $e(\mathbb{G}_1, \mathbb{G}_2) = \mathbb{G}_T$;

KeyGen:

- $s \xleftarrow{\$} \mathbb{Z}_p$ and $v = g_1^s$;
- Set $PK = v \in \mathbb{G}_1$, and $SK = s \in \mathbb{Z}_p$;

Sign($m \in \{0,1\}^*, SK$):

- Return $\sigma \leftarrow H(m)^s$, where $H : \{1,0\}^* \rightarrow \mathbb{G}_2$

Verify($m \in \{0,1\}^*, \sigma, PK$):

- Accept if $e(g_1, \sigma) = e(PK, H(m))$;

Figure 6.4: BLS pairing-based short signature [BLS01].

BGLSAggregateSignature(Aggregate, AggVerify)

Aggregate($\langle m_1, \ldots, m_n \rangle, \langle \sigma_1, \ldots, \sigma_n \rangle$):

- Given that all m_i's are distinct, output $\sigma_a = \prod_{i=1}^{n} \sigma_i$, $\sigma \in \mathbb{G}_2$;

AggVerify($\langle m_1, \ldots, m_n \rangle, \langle PK_1, \ldots, PK_n \rangle, \sigma_a$):

- Reject if m_i's are not distinct;
- $\forall i \in [1, n]$: compute $h_i = H(m_i)$;
- Accept if $e(g_1, \sigma) = \prod_{i=1}^{n} e(PK_i, h_i)$;

Figure 6.5: BGLS aggregate signature [BGLS03].

6.2 Aggregate Signatures

In this section, we propose the use of available aggregate signature schemes as a countermeasure to steganography in blockchains. In 2001, Boneh et al. [BLS01] introduced a short pairing-based signature scheme (BLS) that yields a single group element signature. In particular, BLS uses the following: (1) two multiplicative cyclic groups $\mathbb{G}_1$ and $\mathbb{G}_2$ both of prime order p with g_1 and g_2 as their respective generators, (2) an efficiently computable isomorphism ψ from $\mathbb{G}_1$ to $\mathbb{G}_2$ such that $\psi(g_1) = g_2$, and (3) a computable bilinear map $e : \mathbb{G}_1 \times \mathbb{G}_2 \rightarrow \mathbb{G}_T$. This bilinear mapping has two properties: (3.i) *bi-linearity*: $\forall a \in \mathbb{G}_1, b \in \mathbb{G}_2$ and $r, t \in \mathbb{Z}_p$, $e(a^r, b^t) = e(a, b)^{rt}$, and (3.ii) *non-degeneracy*: $e(g_1, g_2) \neq 1$. Besides, BLS uses a full-domain hash function that maps arbitrary points to $\mathbb{G}_2$, i.e. $H : \{0,1\}^* \rightarrow \mathbb{G}_2$. As shown in Fig. 6.4, the BLS signature is a single element in $\mathbb{G}_2$; $\sigma = H(m)^s \in \mathbb{G}_2$.

It can be seen that BLS is vulnerable to subversion attacks through the rejection sampling of the public keys and, consequently, the generated signature. However, this problem can be mitigated by allowing miners to aggregate the signatures using the bilinear-mapping-based aggregate signature proposed by Boneh et al. [BGLS03], as shown in Fig. 6.5. In this scheme, the messages m_i's are required to be distinct. Nevertheless, as proved by Bellare et al. [BNN07], this limitation can be bypassed by prepending the signer's public key before the message when generating the signature, i.e. $\sigma_i = H(PK_i||m_i)^{s_i}$ instead of $\sigma_i = H(m_i)^{s_i}$. Aggregating signatures according to the scheme shown in Fig. 6.5 or using its unrestricted variant [BNN07] removes any subliminally-leaked data in the aggregated signatures σ_i's. This is attainable because no PPT adversary can re-construct the member aggregated signatures σ_i's given the aggregate signature σ_a. In Appendix C.4, we formally prove that if an adversary $\mathcal{A}$ can dis-aggregate an aggregate signature σ_a, then $\mathcal{A}$ can also solve the hard underlying computational Diffie-Hellman (co-CDH) problem, which is known to be intractable. As such, we say that Boneh et al.'s aggregate signature, as shown in Fig. 6.5, achieves the *dis-aggregation* property, which is necessary to destroy the possible subliminal channels in the member signatures.

Practically, in the context of blockchains, the process of aggregating signatures can only be performed by miners and not other nodes. Thus, using the BGLSAggregateSignature scheme removes arbitrary data from being permanently stored in the blockchain. However, it can not prevent the mere propagation of arbitrary information in the blockchain's P2P network. Consequently, to deter the diffusion of arbitrary content, the blockchain's P2P nodes should be enabled to re-randomize the transactions' signatures to filter out all steganographic information before the signatures are aggregated by the miners.

RAS(KeyGen, Sign, Verify, Randomize, Aggregate, AggVerify)

Given three bilinear groups of type 3: $(\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T)$, prime number p, and bilinear mapping: $e(\mathbb{G}_1, \mathbb{G}_2) = \mathbb{G}_T$;

KeyGen:

- Select $\widetilde{g} \xleftarrow{\$} \mathbb{G}_2$, and $(a, b) \xleftarrow{\$} \mathbb{Z}_p^2$;
- Compute $(\widetilde{A}, \widetilde{B}) \leftarrow (\widetilde{g}^a, \widetilde{g}^b)$;
- $PK = (\widetilde{g}, \widetilde{A}, \widetilde{B}) \in \mathbb{G}_2$, and $SK = (a, b) \in \mathbb{Z}_p$;

Sign(m, SK):

- Select $h \xleftarrow{\$} \mathbb{G}_1 \backslash 1_{\mathbb{G}_1}$;
- Return $\sigma \leftarrow (h, h^{(a+bm)})$;

Verify(PK, m, σ):

- Parse σ as (σ_1, σ_2);
- Accept if $\sigma_1 \neq 1_{\mathbb{G}_1}$, and $e(\sigma_1, \widetilde{A} \cdot \widetilde{B}^m) = e(\sigma_2, \widetilde{g})$;

– –

Randomize(σ):

- Parse σ as (σ_1, σ_2);
- Select $r \xleftarrow{\$} \mathbb{Z}_p^*$;
- Return $\acute{\sigma} \leftarrow (\sigma_1^r, \sigma_2^r)$;

– –

Aggregate$(\langle m_1, \ldots, m_n \rangle, \langle \sigma_1, \ldots, \sigma_n \rangle, \langle PK_1, \ldots, PK_n \rangle)$:

- $\forall i \in [1, n]$: parse σ_i as $(\sigma_{i,1}, \sigma_{i,2})$;
- Proceed only if $\forall i \in [1, n]$:
  - All $\widetilde{g}_i$'s are equal, and
  - All Verify$(PK_i, m_i, \sigma_i) =$ true;
- Return $\sigma_a \leftarrow (\langle \sigma_{(1,1)}, \ldots, \sigma_{(n,1)} \rangle, \prod_{i=1}^{n} \sigma_{(i,2)})$;

AggVerify$(\langle m_1, \ldots, m_n \rangle, \langle PK_1, \ldots, PK_n \rangle, \sigma_a)$:

- Parse σ_a as $(\langle \sigma_{(1,1)}, \ldots, \sigma_{(n,1)} \rangle, \sigma_2)$;
- Accept if $e(\sigma_2, \widetilde{g}) = \prod_{i=1}^{n} e(\sigma_{(i,1)}, \widetilde{A}_i \cdot \widetilde{B}_i^{m_i})$;

Figure 6.6: Randomizable and aggregatable signature (RAS).

6.3 Randomizable and Aggregatable Signature (RAS)

In Sec. 6.2 we have shown that aggregate signatures, such as the BGLSAggregateSignature scheme in Fig. 6.5, can be used by miners to aggregate signatures before adding their respective transactions to mined blocks, and consequently, obfuscate subliminally-embedded content before being persistently stored in the blockchain. Besides, aggregate signatures save storage space in the ledger, ensure that signatures have been re-randomized, and minimize the verification computational cost for subsequent verifiers. However, aggregate signatures do not stop the diffusion of steganographically-communicated information. Therefore, P2P nodes should be enabled to re-randomize the transactions' signatures before they are aggregated by the miners.

On the other hand, Sec. 6.1 shows that randomizable signatures, such as Pointcheval and Sanders's signature in Fig. 6.1, allow the nodes in the P2P network to filter out malicious content and stop it from propagating in the network. Nevertheless, randomizable signatures do not guarantee that miners do not add the original non-re-randomized signatures to their mined blocks, e.g. in the case when a miner receives a signature from the signer directly.

Therefore, in this section, we present a new randomizable and aggregatable signature scheme (RAS) that enables the P2P nodes to re-randomize single signatures, while allowing miners to re-randomize and aggregate the signatures. The RAS scheme consists of six algorithms: (KeyGen, Sign, Verify, Randomize, Aggregate, AggVerify), as shown in Fig. 6.6. The first four algorithms of RAS are the same as in Pointcheval and Sanders's randomizable signature [PS16], as previously discussed in Sec. 6.1. The last two algorithms are inspired by the aggregation in the BGLSAggregateSignature scheme [BGLS03]. The aggregation algorithm Aggregate takes as input n messages $\langle m_1, \ldots, m_n \rangle$ and their corresponding signatures $\langle \sigma_1, \ldots, \sigma_n \rangle$ under their respective n public keys $\langle PK_1, \ldots, PK_n \rangle$. Aggregate starts by parsing each signature σ_i as $(\sigma_{i,1}, \sigma_{i,2})$, then Aggregate verifies each signature σ_i, and if all are valid, it generates an aggregate signature $\sigma_a \leftarrow (\langle \sigma_{(1,1)}, \ldots, \sigma_{(n,1)} \rangle, \prod_{i=1}^{n} \sigma_{(i,2)})$. When receiving an aggregate signature σ_a with its signed messages $\langle m_1, \ldots, m_n \rangle$ and corresponding public keys, the aggregate

verification algorithm AggVerify parses σ_a as $(\langle\sigma_{(1,1)},\ldots,\sigma_{(n,1)}\rangle,\sigma_2)$, and accepts the aggregate signature is valid only if $e(\sigma_2,\widetilde{g}) = \prod_{i=1}^{n} e(\sigma_{(i,1)},\widetilde{A}_i \cdot \widetilde{B}_i^{m_i})$. The correctness of RAS is given as follows:

$$e(\sigma_2,\widetilde{g}) = e(\prod_{i=1}^{n}\sigma_{(i,2)},\widetilde{g}) = e(\prod_{i=1}^{n} h_i^{(a_i+b_i\cdot m_i)r_i},\widetilde{g}) = \prod_{i=1}^{n} e(h_i^{(a_i+b_i\cdot m_i)r_i},\widetilde{g})$$
$$= \prod_{i=1}^{n} e(h_i^{r_i},\widetilde{g}^{(a_i+b_i\cdot m_i)}) = \prod_{i=1}^{n} e(\sigma_{(i,1)},A_i\cdot B_i^{m_i})$$

Security of RAS: RAS comprises of two parts. (i) The first four algorithms (KeyGen, Sign, Verify, Randomize) are the same as Pointcheval and Sanders's randomizable signature [PS16]. (ii) The last two algorithms are similar to the BGLSAggregateSignature in Fig. 6.5. As such, the security proof of RAS in Appendix C.5 starts by describing the proof of Pointcheval and Sanders's re-randomizable signature [PS16], and then we prove that our aggregated signature scheme is existentially unforgeable under chosen-message attacks (EUF-CMA) based on the security of Pointcheval and Sanders's signature. Specifically, we use reduction to prove that if there is any probabilistic polynomial-time (PPT) adversary who can forge a RAS aggregate signature σ_a with a non-negligible probability, then there must exist another PPT adversary who breaks the security of Pointcheval and Sanders's randomizable signature and its underlying security assumption.

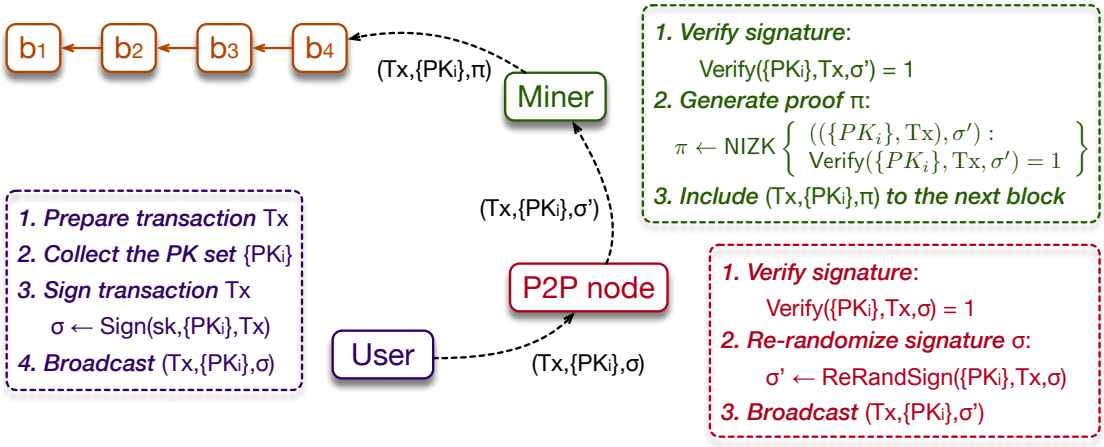


Figure 6.7: Steganography-resistant blockchain framework (SRBF)

6.4 Steganography-Resistant Blockchain Framework

In this section, we propose a universal steganography-resistant blockchain framework (SRBF) that can be deployed on any blockchain system. Without loss of generality, we explain our technique to specifically defend against kleptographic and steganographic attacks on signature schemes; however, it can be applied analogously to other cryptographic components of the blockchain, such as the non-interactive zero-knowledge (NIZK) proofs. The SRBF framework operates on the assumption that any sender of a transaction may be maliciously implemented to exploit the random cryptographic signatures. As depicted in Fig. 6.7, to immunize blockchains against steganography, the proposed SRBF introduces two elements in the form of new duties that are assigned to the P2P nodes and the miners.

Duty of P2P nodes in SRBF. The first element of SRBF is to require P2P nodes to sanitize and re-randomize signatures. Currently, the P2P nodes in blockchains, e.g. in Bitcoin [KKM14], check the validity of broadcast transactions and their signatures before relaying them into the network. In SRBF, P2P nodes also re-randomize the received signatures before propagating broadcast transactions. This practice filters out any possible steganographically-embedded data and necessitates the use of re-randomizable signatures like Pointcheval and Sanders's randomizable signature [PS16]. Besides, this approach removes the need for reverse firewalls [AMV15] and interactive wardens [ZLLZ13].

Duty of miners in SRBF. The second element of the SRBF framework is to require the miners to replace the signatures with NIZK proofs. Conventionally, upon receiving a transaction tx, the miners would check the validity of its associated signature(s) σ, using the signature verification algorithm $\mathsf{Verify}(\mathrm{PK}, \mathsf{tx}, \sigma) = 1$. The miners then include the transaction together with its signature as it is to the next block, which will eventually be appended to the blockchain. Afterwards, other miners and users can verify the validity of the transaction as well. However, in the SRBF framework, the miner, instead of showing the signature, replaces the transaction's signature with a NIZK proof. Informally, the proof states that 'I have seen a valid signature such that $\mathsf{Verify}(\mathrm{PK}, \mathsf{tx}, \sigma) = 1$'. More

precisely, the NIZK proof proves the following relation:

$$\mathcal{R}_{\mathsf{sig}} = \{((\{\mathrm{PK}_i\}_{i=1}^{n}, \mathsf{tx}), \sigma) | \mathsf{Verify}(\{\mathrm{PK}_i\}_{i=1}^{n}, \mathsf{tx}, \sigma) = 1\}$$

Thus, in the SRBF framework, only the transaction tx, its signer(s) public key(s) $\{\mathrm{PK}_i\}_{i=1}^{n}$, and its NIZK proof π will be posted on the blockchain, where π is given as follows:

$$\pi \leftarrow \mathsf{NIZK} \left\{ \begin{array}{l} ((\{\mathrm{PK}_i\}_{i=1}^{n}, \mathsf{tx}), \sigma) : \\ \mathsf{Verify}(\{\mathrm{PK}_i\}_{i=1}^{n}, \mathsf{tx}, \sigma) = 1 \end{array} \right\}$$

Alternatively, to avoid the computational cost incurred by the use of NIZK proofs, miners can use aggregate signatures, as previously explained in Sec. 6.2 and Sec. 6.3. For example, a miner can use Boneh et al.'s aggregate signature [BGLS03] to generate an aggregate signature σ_a for n signatures on n distinct messages, i.e. $(\langle m_1, \ldots, m_n \rangle, \langle \sigma_1, \ldots, \sigma_n \rangle)$. The aggregate signature in this specific example is given as $\sigma_a = \prod_{i=1}^{n} \sigma_i$.

While the SRBF framework increases the computational effort for miners, it drastically decreases the effort for others to verify the validity of a given block; hence, mitigating the verifier's dilemma issue [LTKS15], where multiple transactions' signatures are combined in one NIZK proof π, or aggregated in one aggregate signature σ_a.

Implementation of the SRBF in cryptocurrencies. On the one hand, implementing the SRBF in practice faces two challenges. First, it necessitates finding efficient mechanisms to incentivize the P2P nodes to actively participate and randomize the transactions before propagating them into the network. The second challenge is manifested by the additional computational effort by the miners to generate the zero-knowledge proofs or the aggregate signatures.

On the other hand, the SRBF can be implemented gradually in phases and partially which makes it practically achievable. Gradual implementation of the SRBF implies that the duty of P2P nodes, as aforementioned, can be implemented and tested before implementing the duty of the miners. Whereas, the partial implementation allows the SRBF's features to be introduced in subsets of the nodes and can initially be supported

concurrently with older features. When the new features are deemed to be stable, the older features can be redacted and no longer supported. A similar approach of the partial and gradual introduction of new security features is followed by Monero as illustrated in Sec. A.2.6 of Appendix A.

Security of the SRBF. P2P nodes obstruct kleptographic attacks and steganographic communication by re-randomizing the signatures. Additionally, if miners use NIZK proofs, then the signature σ is the witness of the corresponding proof π . By NIZK definition, π does not leak any information about σ. Therefore, all the steganographic information that may be hidden in the signatures is filtered out from the blockchain. In practice, we can use, for example, Bulletproofs [BBB$^+$18] as succinct NIZK instantiation.

On the other hand, when miners use aggregate signatures instead of using NIZK proofs, given that σ_a achieves the *dis-aggregation* property as explained in Sec. 6.2, there is not any PPT adversary that can re-construct the member signatures σ_i's. Consequently, similar to replacing signatures with NIZK proofs, using aggregate signatures effectively obliterates any possibly steganographic content in the aggregated signatures.

Technique	Attack Scenarios			Input Trigger	Practical in blockchains	References	Section
	Kleptography	Steganography					
	Subversion attacks	Covert communication	Persistent storage				
Steganography-resistant techniques							
De-randomized algorithms	✓	χ	χ	χ	✓	[YY96]	5.1.1
Cascading cryptosystems	✓	χ	χ	✓	χ	[YY96]	5.1.2
Software integrity	✓	χ	χ	✓	χ	[YY96]	5.1.3
Synthetically-random primitives	✓	✓	✓	χ	χ	[BPR14] [BJK15]	5.1.4
VRFs and controlled randomness	χ	χ	χ	✓	χ	[MRV99] [HKK17]	5.1.5
Split model	✓	χ	χ	χ	χ	[RTYZ16a] [RTYZ16b] [RTYZ17]	5.1.6
Interactive warden	✓	✓	✓	✓	χ	[ZLLZ13]	5.1.7
Reverse firewalls	✓	✓	✓	✓	χ	[AMV15]	5.1.8
Self-guarding protocols	✓	χ	χ	✓	χ	[FM18]	5.1.9
Current blockchain-based techniques							
Lightweight blockchains	χ	χ	✓	✓	✓	[MS17] [J.D17]	5.2.1
Prunable blockchains	χ	χ	✓	✓	✓	[FHBS19] [CLO16]	5.2.2
Redactable blockchains	χ	χ	✓	✓	✓	[AMVA17] [PDC17]	5.2.3
Fees, filters and self-verifying addresses	χ	χ	χ	χ	✓	[MHZ+18]	5.2.4
Proposed countermeasures							
Re-randomizable cryptographic primitives	✓	✓	✓	✓	✓	[PS16]†	6.1
RRS	✓	✓	✓	✓	✓	this work	6.1.1
Aggregate signatures	χ	χ	✓	✓	✓	[BGLS03]†	6.2
RAS	✓	✓	✓	✓	✓	this work	6.3
SRBF	✓	✓	✓	✓	✓	this work	6.4

Table 6.1: Effectiveness of proposed and current countermeasures against the three attack scenarios (Chapters 3 and 4). (✓) denotes that the relevant countermeasure is resistant against the corresponding attack scenario, while (χ) means the countermeasure is vulnerable to the attack scenario. (*Input-trigger*) states if the countermeasure is resistant to 'time bombs' or input-triggered malicious behaviour. Finally, (*Practical in blockchains*) examines if the corresponding countermeasure is applicable in the context of blockchains, if it is not likely to produce other security ramifications, and its robustness against malicious users. †: the cited references proposed these cryptographic schemes; however, no previous work has suggested their use to deter kleptography and steganography in the context of blockchains.

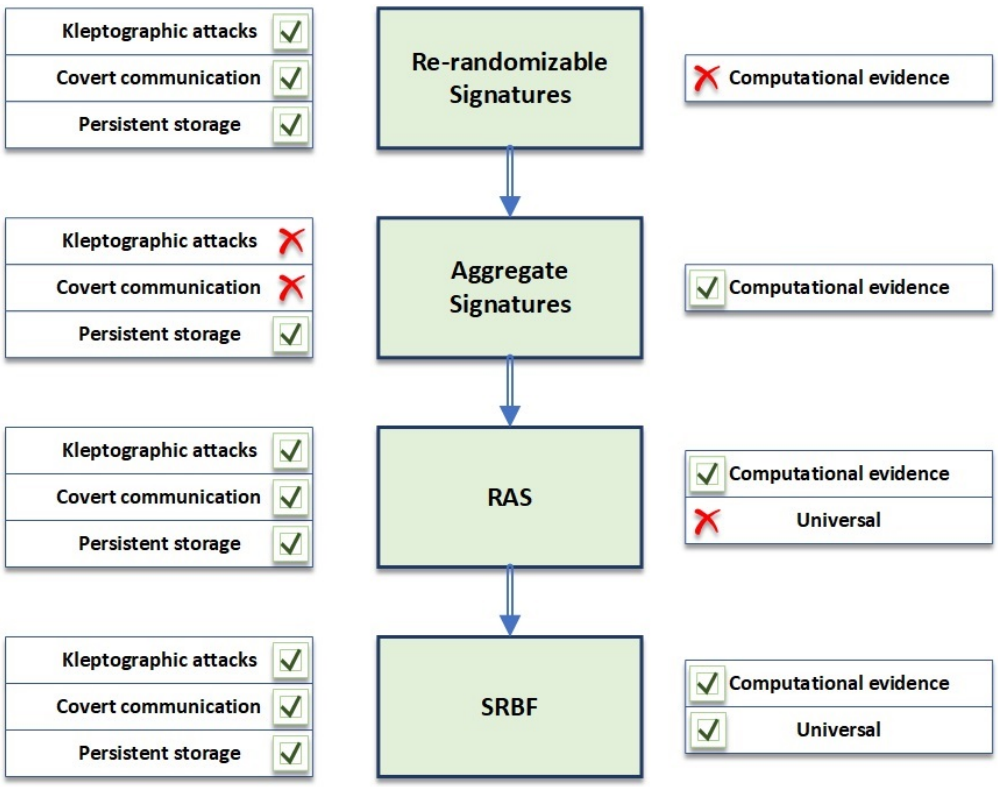


Figure 6.8: Graphical comparison between the proposed countermeasures. Re-randomizable signatures defeat all three attack scenarios but do not provide a 'computational evidence' that a signature have been re-randomized. Aggregate signatures provide evidence but fail to deter kleptographic attacks and covert communication. RAS is efficient in countering all three attack scenarios but it is not universal. SRBF presents a universal framework that defeats all attack scenarios.

6.5 Summary of Countermeasures

Chapter 5 surveyed all the cryptographic techniques that have been proposed to defeat kleptography and subliminal channels, and examined novel blockchain designs in light of the attack scenarios of Chapters 3 and 4. Notably, as shown in Table 6.1, some of the existing cryptographic primitives are efficient deterrence against kleptography; nevertheless, they fail to prevent steganography and vice versa. Nonetheless, three existing techniques can obstruct all attack scenarios: (1) synthetically-random primitives [BPR14, BJK15], (2) interactive wardens [ZLLZ13], and (3) cryptographic reverse firewalls [AMV15]. However, all of these techniques are deemed to be impractical in the context of public blockchains for the following reasons. Synthetically-random primitives are susceptible to input-triggered attacks. Whereas, the last two techniques defeat the decentralized design of blockchains, and require the existence of trusted parties in the form of wardens and firewalls.

Due to the shortcomings of all of the existing techniques, this chapter proposed four new countermeasures: (1) re-randomizable signatures, (2) aggregate signatures, (3) re-randomizable and aggregatable signatures, and (4) a generic steganography-resistant blockchain framework (SRBF). As summarized in Table 6.1, the use of re-randomizable signatures is effective against kleptography and steganography. However, re-randomizable signatures do not provide any computational evidence that a signature had been re-randomized before it is included in a mined block. In contrast, aggregate signatures can prevent the permanent storage of arbitrary content in blockchain, yet, they can not deter kleptographic attacks on blockchains, nor can they stop covert communication. To address the shortcomings of the first two countermeasures, we proposed a new cryptographic re-randomizable and aggregatable signature (RAS). Finally, we presented SRBF as a universal framework that thwarts kleptography and steganography in public blockchains regardless of the specific used cryptographic primitives. For further illustration, Fig. 6.8 presents a graphical comparison between the proposed countermeasures.

Chapter 7

Conclusion and Future Work

7.1 Conclusion

The blockchain technology has presented a significant paradigm shift concerning the realization of decentralized applications. Public blockchains are commonly developed as open-source projects, and their clients can be arbitrarily modified by the end-users while seamlessly interacting with other unmodified clients. As demonstrated in this thesis, these modifications may be designed to kleptographically leak the users' secret information. Also, users may knowingly modify their clients to communicate steganographically or store malicious content in public blockchains. Nonetheless, public blockchains have been designed with disregard to the adverse effects of such malicious modifications. Consequently, this thesis's primary goal is to highlight the threat of kleptographic and steganographic attacks on public blockchains and research novel ways to prevent them.

About twenty-five years ago, Young and Yung pointed out the danger of trusting black-box cryptography [YY96, YY97a, YY97b] and coined the term 'kleptography'. Subsequent work has confirmed that randomized cryptographic primitives are inevitably vulnerable to kleptographic and algorithm-substitution attacks [BPR14, BJK15]. Yet, this issue has not attracted sufficient research in the context of public blockchains, and current clients of public blockchain applications seem to have been designed with the

assumption that users, who can not scrutinize the implementation for themselves, should blindly trust the cryptography of their clients. Besides, despite the known susceptibility of randomized channels to steganography, before our work, there had not been extensive research on the detrimental effects of steganographic communication through public blockchains.

To shed light on the risk posed by the mis-implementation of the randomized cryptographic primitives in public blockchains, we conducted the following. (1) We proposed three kleptographic attacks on two commonly used randomized digital signature schemes. (2) We studied the impact of steganographic communication on public blockchains and devised an economically-efficient and provably-secure broadcast communication application called Skywhisper. (3) We surveyed and assessed all of the possible techniques that can mitigate kleptography and steganography in public blockchains. (4) Due to the inefficiency of the current techniques, we proposed four new countermeasures to ensure steganography-resistant blockchains. In the following paragraphs, we summarize each of these four points.

To demonstrate kleptographic attacks on public blockchains, we proposed two attacks on the ECDSA signature and one attack on CryptoNote's ring signature. The first attack on ECDSA leaks the user's secret signing key over two consecutive signatures. This is attained by choosing the second signature's ephemeral randomness synthetically based on the first signature's randomness and the attacker's public key. This attack has a high throughput; however, it can only be used to leak the signing key and not any other confidential information. Besides, it is stateful; hence, it can be thwarted if the software is reset every time a signature is generated. In contrast, our second attack on ECDSA is a stateless attack based on the rejection sampling of the signature's ephemeral randomness, and it can be used to leak arbitrary information bit by bit. Also, our attack on the ring signature offers a high throughput that is proportional to the size of the signature. Overall, although with varying bandwidth, all of these three attacks can be used to exfiltrate the users' secret signing key, which is the most valuable secret in public blockchains, and particularly in cryptocurrencies, leading to the potential theft of the

users' cryptocurrency funds.

Moreover, we explored the severe ramifications of steganographic communication on blockchains and clarified that malicious users can abuse public blockchains as uncensorable covert communication platforms and secure cyberlockers. We argue that abusing blockchains to persistently store objectionable content does not only infringe the right to be forgotten (RtbF) but poses a regulatory risk. It can also motivate users to abandon the blockchain technology or at least opt-out of the participation in the consensus protocol. To enhance the economic feasibility of steganographic communication in public blockchains, malicious users can target cryptocurrencies, whose transaction fees are minimal and offer high throughput per transaction. Given the known rapport between secure steganographic communication and undetectable kleptographic attacks [BL17], we used our kleptographic attack on CryptoNote's ring signature to realize a steganographic broadcast communication tool, called Skywhisper, over the real-world blockchain of Bytecoin. Skywhisper offers a provably-secure broadcast channel of up to 64 subscribers. In Skywhisper, a transaction of 4 inputs and 10 public keys can transmit more than 2KB of covert data and costs \$0.0000215.

To deter kleptographic attacks and prevent steganographic communication on public blockchains, we investigated all of the available cryptographic countermeasures and new blockchain designs. Moreover, we assessed the effectiveness of each method in countering three attack scenarios: (1) kleptographically leaking the users' secret information, (2) steganographically communicating over public blockchains, and (3) persistently storing arbitrary content in blockchains. All of the surveyed techniques fail to counter at least one of the aforementioned scenarios or are deemed impractical in the context of blockchains.

Consequently, to fully immunize public blockchains against kleptography and steganography, we presented four countermeasures. The first is to use re-randomizable signature schemes, which enable the blockchain P2P nodes to re-randomize the transactions' signatures before broadcasting them onto the network. Besides, we presented a new re-randomizable ring signature (RRS) with provable security. The process of re-randomizing the signature also filters out any possible subliminal channels. However,

since re-randomized signatures are computationally indistinguishable from the original signatures, there is not any computational evidence that a signature has ever been re-randomized before its respective transaction is added to a mined block. To address this shortcoming, our second countermeasure is to use aggregate signatures, where miners aggregate signatures of independent transactions, and only the aggregated signature is added with the corresponding transactions to the mined block. Nonetheless, aggregate signatures do not prevent the mere propagation of steganographic communication in the P2P network. Therefore, we presented our third countermeasure, which is to use re-randomizable and aggregatable signatures. Additionally, we devised a new cryptographic primitive called re-randomizable and aggregatable signature (RAS). The fourth countermeasure is a universal steganography-resistant blockchain framework (SRBF), which ensures steganography resistance in blockchains. The SRBF framework depends on the use of re-randomizable signatures and either non-interactive zero-knowledge proofs (NIZK) or aggregate signatures.

This thesis highlights the imminent threat that kleptography and steganography pose on public blockchains' users and technology. Our aims have been achieved by designing realistic kleptographic attacks and steganographic channels on two of the most widely used signature schemes in blockchains. In addition, this thesis has introduced a set of countermeasures to design steganography-resistant blockchains. Finally, this thesis opens new research venues and serves to instigate further efforts to immunize public blockchains against subversion attacks on randomized cryptographic primitives and design kleptography and steganography-resistant blockchains.

7.2 Future Work

This thesis has explored the threat of kleptography and steganography on public blockchains, investigated current deterrence techniques, and proposed four novel and efficient countermeasures. However, this thesis also serves to instigate further interest in the subject, encourage the blockchain community to consider the adverse effects of kleptography and steganography, and inspire new steganographic-resistant blockchains.

As explained in Chapter 4, abusing public blockchains for steganographic communication and content storage jeopardizes the future proliferation of the blockchain technology. More specifically, out of fear of legal prosecution or from a moral standpoint, users may avoid using blockchains that are known to contain malicious content. Users who choose to participate in such blockchains may refrain from downloading the full ledger, and thus stop engaging in the consensus protocol, which leads to fewer nodes participating in the process. Since the consensus protocol is at the crux of the security in public blockchains and represents the primary defence against adverse attacks such as denial-of-service and double-spending attacks, more research is required to quantify the effect of such behaviour on the overall security of blockchains.

Furthermore, this research has proposed two new cryptographic signatures: the randomizable ring signature (RRS) and the re-randomizable and aggregatable signature (RAS). However, beyond the proposed cryptographic signatures, this work serves to encourage the cryptography community to design new steganography-resistant randomized cryptographic primitives. Also, more computationally-efficient alternatives can be considered to realize the SRBF framework instead of using the computationally-intensive non-interactive zero-knowledge proofs (NIZK) and aggregate signatures.

In addition, theoretically, the proposed countermeasures in this thesis can be applied to resist steganography and kleptography in other open-source applications beyond their mere applicability in blockchains. Nonetheless, a thorough security analysis of each applicable scenario is required to assess their security and evaluate their practical applicability.

Finally, our proposed steganography-resistant blockchain framework (SRBF) provides the basis for future steganography-resistant blockchain designs. Nevertheless, further work is required to implement the SRBF framework on an open-source blockchain platform, such as HyperLedger[1], investigate ways to incentivize nodes to re-randomize signatures, and practically evaluate the SRBF framework. This effort will help demonstrate the practical effectiveness of SRBF against kleptography and steganography.

[1] `https://www.hyperledger.org/`

Bibliography

[A. 15] A. Mackenzie, S. Noether and Monero Core Team. Improving Obfuscation in the CryptoNote Protocol, 1 2015. Available Online: `https://lab.getmonero.org/pubs/MRL-0004.pdf` (Last accessed 01-Jan.-2019). [Cited on page 144]

[ACST06] Man Ho Au, Sherman S. M. Chow, Willy Susilo, and Patrick P. Tsang. Short linkable ring signatures revisited. In Andrea S. Atzeni and Antonio Lioy, editors, *Public Key Infrastructure*, pages 101–115, Berlin, Heidelberg, 2006. Springer Berlin Heidelberg. [Cited on page 19]

[ADR02] Jee Hea An, Yevgeniy Dodis, and Tal Rabin. On the security of joint signature and encryption. In Lars R. Knudsen, editor, *Advances in Cryptology — EUROCRYPT 2002*, pages 83–107, Berlin, Heidelberg, 2002. Springer Berlin Heidelberg. [Cited on pages 16, 17, 141, and 142]

[AEVL16] 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)*, pages 25–30, 2016. [Cited on page 1]

[AKR+13] Elli Androulaki, Ghassan O. Karame, Marc Roeschlin, Tobias Scherer, and Srdjan Capkun. Evaluating user privacy in bitcoin. In Ahmad-Reza Sadeghi, editor, *Financial Cryptography and Data Security*, pages 34–51, Berlin, Heidelberg, 2013. Springer Berlin Heidelberg. [Cited on pages 137 and 138]

[AMM18] Sarah Azouvi, Mary Maller, and Sarah Meiklejohn. Egalitarian society or benevolent dictatorship: The state of cryptocurrency governance. In *5th Workshop on Bitcoin and Blockchain Research*, 2018. [Cited on pages 4 and 37]

[AMV15] Giuseppe Ateniese, Bernardo Magri, and Daniele Venturi. Subversion-resilient signature schemes. In *CCS '15*, pages 364–375, 2015. [Cited on pages 4, 6, 15, 22, 25, 26, 27, 32, 35, 84, 87, 96, 104, 107, and 109]

[AMVA17] G. Ateniese, B. Magri, D. Venturi, and E. Andrade. Redactable blockchain – or – rewriting history in bitcoin and friends. In *Euro S&P 2017*, pages 111–126, April 2017. [Cited on pages 31, 62, 92, and 107]

[And96] Ross Anderson. Stretching the limits of steganography. In *Information Hiding*, pages 39–48. Springer Berlin Heidelberg, 1996. [Cited on pages 6 and 27]

[AOS02] Masayuki Abe, Miyako Ohkubo, and Koutarou Suzuki. 1-out-of-n signatures from a variety of keys. In *ASIACRYPT 2002*, 2002. [Cited on pages 21 and 53]

[AP98] R. J. Anderson and F. A. P. Petitcolas. On the limits of steganography. *IEEE Journal on Selected Areas in Communications*, 16(4):474–481, May 1998. [Cited on pages 6 and 27]

[AZ19a] N. Alsalami and B. Zhang. Sok: A systematic study of anonymity in cryptocurrencies. In *2019 IEEE Conference on Dependable and Secure Computing (DSC)*, pages 1–9, Nov 2019. [Cited on pages 9, 10, and 135]

[AZ19b] N. Alsalami and B. Zhang. Utilizing public blockchains for censorship-circumvention and iot communication. In *2019 IEEE Conference on Dependable and Secure Computing (DSC)*, pages 1–7, Nov 2019. [Cited on page 61]

[BB04] Dan Boneh and Xavier Boyen. Short signatures without random oracles. In Christian Cachin and Jan L. Camenisch, editors, *Advances in Cryptology - EUROCRYPT 2004*, pages 56–73, Berlin, Heidelberg, 2004. Springer Berlin Heidelberg. [Cited on page 17]

[BBB+18] B. Bünz, J. Bootle, D. Boneh, A. Poelstra, P. Wuille, and G. Maxwell. Bulletproofs: Short proofs for confidential transactions and more. In *2018 IEEE Symposium on Security and Privacy (SP)*, pages 315–334, May 2018. [Cited on pages 53, 106, 140, and 143]

[BBC19] BBC. Child abuse images hidden in crypto-currency blockchain, 2019. Available Online: `https://www.bbc.co.uk/news/technology-47130268` (Last accessed 02-August-2019). [Cited on pages 5 and 61]

[BBG13] James Ball, Julian Borger, and Glenn Greenwald. Revealed: how US and UK spy agencies defeat internet privacy and security, September 2013. Available Online: `https://www.theguardian.com/world/2013/sep/05/nsa-gchq-encryption-codes-security` (Last accessed 12-June-2020). [Cited on pages 4 and 24]

[BCC+16] Jonathan Bootle, Andrea Cerulli, Pyrros Chaidos, Jens Groth, and Christophe Petit. Efficient zero-knowledge arguments for arithmetic circuits in the discrete log setting. In Marc Fischlin and Jean-Sébastien Coron, editors, *Advances in Cryptology – EUROCRYPT 2016*, pages 327–357, Berlin, Heidelberg, 2016. Springer Berlin Heidelberg. [Cited on page 143]

[BDL+11] Daniel J. Bernstein, Niels Duif, Tanja Lange, Peter Schwabe, and Bo-Yin Yang. High-speed high-security signatures. In Bart Preneel and Tsuyoshi Takagi, editors, *Cryptographic Hardware and Embedded Systems – CHES 2011*, pages 124–142, Berlin, Heidelberg, 2011. Springer Berlin Heidelberg. [Cited on pages 26 and 84]

[BDP+15] G. D. Battista, V. D. Donato, M. Patrignani, M. Pizzonia, V. Roselli, and R. Tamassia. Bitconeview: visualization of flows in the bitcoin transaction graph. In *2015 IEEE Symposium on Visualization for Cyber Security (VizSec)*, pages 1–8, Oct 2015. [Cited on page 139]

[bes19] BestMixer, 2019. Available Online: `https://bestmixer.io` (Last accessed 22-Feb-2019). [Cited on page 141]

[BGLS03] Dan Boneh, Craig Gentry, Ben Lynn, and Hovav Shacham. Aggregate and verifiably encrypted signatures from bilinear maps. In Eli Biham, editor, *Advances in Cryptology — EUROCRYPT 2003*, pages 416–432, Berlin, Heidelberg, 2003. Springer Berlin Heidelberg. [Cited on pages 97, 98, 99, 100, 102, 105, 107, 148, 152, 153, and 155]

[BGVS07] Jens-Matthias Bohli, María Isabel González Vasco, and Rainer Steinwandt. A subliminal-free variant of ecdsa. In Jan L. Camenisch, Christian S. Collberg, Neil F. Johnson, and Phil Sallee, editors, *Information Hiding*, pages 375–387, Berlin, Heidelberg, 2007. Springer Berlin Heidelberg. [Cited on pages 27 and 85]

[BGW05] D. Boneh, C. Gentry, and B. Waters. Collusion resistant broadcast encryption with short ciphertexts and private keys. In Victor Shoup, editor, *Advances in Cryptology – CRYPTO 2005*, pages 258–275, Berlin, Heidelberg, 2005. Springer Berlin Heidelberg. [Cited on pages 70, 75, and 77]

[BH15] Mihir Bellare and Viet Tung Hoang. Resisting randomness subversion: Fast deterministic and hedged public-key encryption in the standard model. In Elisabeth Oswald and Marc Fischlin, editors, *Advances in Cryptology - EUROCRYPT 2015*, pages 627–656, Berlin, Heidelberg, 2015. Springer Berlin Heidelberg. [Cited on pages 25, 32, and 84]

[bit19a] Bitcoin Fog, 2019. Available Online: `https://bitcoinfog.info` (Last accessed 12-Aug-2019). [Cited on page 141]

[bit19b] Bitcoin Project, 2019. Available Online: `https://bitcoin.org/en/` (Last accessed 22-Aug-2019). [Cited on page 137]

[BJK15] Mihir Bellare, Joseph Jaeger, and Daniel Kane. Mass-surveillance without the state: Strongly undetectable algorithm-substitution attacks. In *Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security*, CCS '15, page 1431–1440, New York, NY, USA, 2015. Association for Computing Machinery. [Cited on pages 4, 6, 22, 25, 26, 32, 84, 107, 109, and 110]

[BKP14] Alex Biryukov, Dmitry Khovratovich, and Ivan Pustogarov. Deanonymisation of clients in bitcoin p2p network. In *Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security*, CCS '14, pages 15–29, New York, NY, USA, 2014. ACM. [Cited on pages 139 and 145]

[BL17] Sebastian Berndt and Maciej Liśkiewicz. Algorithm substitution attacks from a steganographic perspective. In *Proceedings of the 2017 ACM SIGSAC*

Conference on Computer and Communications Security, CCS '17, pages 1649–1660, New York, NY, USA, 2017. ACM. [Cited on pages 36, 37, 62, 68, 79, and 112]

[BLS01] Dan Boneh, Ben Lynn, and Hovav Shacham. Short signatures from the weil pairing. In Colin Boyd, editor, *Advances in Cryptology — ASIACRYPT 2001*, pages 514–532, Berlin, Heidelberg, 2001. Springer Berlin Heidelberg. [Cited on pages 97, 99, and 100]

[BNM+14] Joseph Bonneau, Arvind Narayanan, Andrew Miller, Jeremy Clark, Joshua A. Kroll, and Edward W. Felten. Mixcoin: Anonymity for bitcoin with accountable mixes. In Nicolas Christin and Reihaneh Safavi-Naini, editors, *Financial Cryptography and Data Security*, pages 486–504, Berlin, Heidelberg, 2014. Springer Berlin Heidelberg. [Cited on page 141]

[BNN07] Mihir Bellare, Chanathip Namprempre, and Gregory Neven. Unrestricted aggregate signatures. In Lars Arge, Christian Cachin, Tomasz Jurdziński, and Andrzej Tarlecki, editors, *Automata, Languages and Programming*, pages 411–422, Berlin, Heidelberg, 2007. Springer Berlin Heidelberg. [Cited on page 100]

[BP15] Alex Biryukov and Ivan Pustogarov. Bitcoin over tor isn't a good idea. In *2015 IEEE Symposium on Security and Privacy*, pages 122–134, Washington, DC, USA, 2015. IEEE Computer Society. [Cited on pages 139 and 145]

[BPR14] Mihir Bellare, Kenneth G. Paterson, and Phillip Rogaway. Security of symmetric encryption against mass surveillance. In *CRYPTO 2014*, pages 1–19, Berlin, Heidelberg, 2014. Springer Berlin Heidelberg. [Cited on pages 4, 6, 22, 24, 26, 32, 33, 34, 36, 84, 107, 109, and 110]

[BSKC19] J. Baek, W. Susilo, J. Kim, and Y. Chow. Subversion in practice: How to efficiently undermine signatures. *IEEE Access*, 7:68799–68811, 2019. [Cited on page 26]

[BSS02] Emmanuel Bresson, Jacques Stern, and Michael Szydlo. Threshold ring signatures and applications to ad-hoc groups. In Moti Yung, editor, *Advances in Cryptology — CRYPTO 2002*, pages 465–480, Berlin, Heidelberg, 2002. Springer Berlin Heidelberg. [Cited on page 19]

[BSW11] Elette Boyle, Gil Segev, and Daniel Wichs. Fully leakage-resilient signatures. In *EUROCRYPT 2011*, 2011. [Cited on page 35]

[But16] Vitalik Buterin. Privacy on the blockchain, 2016. Available Online : `https://blog.ethereum.org/2016/01/15/privacy-on-the-blockchain/`(Last accessed 09-Oct.-2018). [Cited on page 137]

[Byt18] Bytecoin Org. Bytecoin (bcn), 2018. Available Online: `https://bytecoin.org/` (Last accessed 17-June-2020). [Cited on pages 8, 10, 11, 19, and 37]

[Cac98] Christian Cachin. An information-theoretic model for steganography. In *Information Hiding*, 1998. [Cited on pages 27, 28, and 29]

[CDK+17] Jan Camenisch, David Derler, Stephan Krenn, Henrich C. Pöhls, Kai Samelin, and Daniel Slamanig. Chameleon-hashes with ephemeral trapdoors. In *PKC 2017*, 2017. [Cited on page 92]

[CFN90] David Chaum, Amos Fiat, and Moni Naor. Untraceable electronic cash. In Shafi Goldwasser, editor, *Advances in Cryptology — CRYPTO' 88*, pages 319–327, New York, NY, 1990. Springer New York. [Cited on page 13]

[CH91] David Cham and E. van Heyst. Group signatures. In D. W. Davies, editor, *Eurocrypt 1991*, pages 257–65. Springer-Verlag, 1991. [Cited on page 18]

[Cha83] David Chaum. Blind signatures for untraceable payments. In David Chaum, Ronald L. Rivest, and Alan T. Sherman, editors, *Advances in Cryptology*, pages 199–203, Boston, MA, 1983. Springer US. [Cited on page 13]

[CLO16] Alexander Chepurnoy, Mario Larangeira, and Alexander Ojiganov. Rollerchain, a blockchain with safely pruneable full blocks, 2016. [Cited on pages 91 and 107]

[Coi18] CoinMarketCap. Cryptocurrency market capitalizations, 2018. Available Online: `https://coinmarketcap.com/` (Last accessed 26-Nov-2018). [Cited on pages 76 and 77]

[Cop96] Don Coppersmith. Finding a small root of a bivariate integer equation; factoring with high bits known. In Ueli Maurer, editor, *Advances in Cryptology — EUROCRYPT '96*, pages 178–189, Berlin, Heidelberg, 1996. Springer Berlin Heidelberg. [Cited on page 25]

[cry19] CryptoMixer, 2019. Available Online: `https://cryptomixer.io/` (Last accessed 15-Aug-2019). [Cited on page 141]

[CS03] Claude Crépeau and Alain Slakmon. Simple backdoors for rsa key generation. In Marc Joye, editor, *Topics in Cryptology — CT-RSA 2003*, pages 403–416, Berlin, Heidelberg, 2003. Springer Berlin Heidelberg. [Cited on page 25]

[CSLR18] M. Conti, E. Sandeep Kumar, C. Lal, and S. Ruj. A survey on security and privacy issues of bitcoin. *IEEE Communications Surveys Tutorials*, 20(4):3416–3452, 2018. [Cited on pages 65 and 135]

[Cup20] Miron Cuperman. Gitian, 2020. Available Online: `https://gitian.org/` (Last accessed 20-May-2020). [Cited on page 83]

[CXT06] Changyong Xu, Xijian Ping, and Tao Zhang. Steganography in compressed video stream. In *First International Conference on Innovative Computing, Information and Control - Volume I (ICICIC'06)*, volume 1, pages 269–272, 2006. [Cited on page 29]

[DA99] T. Dierks and C. Allen. The TLS Protocol - Version 1.0. RFC 2246, RFC Editor, January 1999. Available online: `https://tools.ietf.org/html/rfc2246` (Last accessed 09-April-2019). [Cited on page 25]

[DFP15] Jean Paul Degabriele, Pooya Farshim, and Bertram Poettering. A more cautious approach to security against mass surveillance. In Gregor Leander, editor, *Fast Software Encryption*, pages 579–598, Berlin, Heidelberg, 2015. Springer Berlin Heidelberg. [Cited on pages 25 and 84]

[DGG+15] Yevgeniy Dodis, Chaya Ganesh, Alexander Golovnev, Ari Juels, and Thomas Ristenpart. A formal treatment of backdoored pseudorandom generators. In Elisabeth Oswald and Marc Fischlin, editors, *Advances in Cryptology – EUROCRYPT 2015*, pages 101–126, Berlin, Heidelberg, 2015. Springer Berlin Heidelberg. [Cited on pages 26 and 86]

[DH76] W. Diffie and M. Hellman. New directions in cryptography. *IEEE Transactions on Information Theory*, 22(6):644–654, November 1976. [Cited on page 144]

[DI18] Docker-Inc. Docker, 2018. Available Online: `https://www.docker.com` (Last accessed 7-Feb-2018). [Cited on page 86]

[DIRR09] Nenad Dedić, Gene Itkis, Leonid Reyzin, and Scott Russell. Upper and lower bounds on black-box steganography. *Journal of Cryptology*, 22(3):365–394, Jul 2009. [Cited on pages 58 and 61]

[DMSD16] Yevgeniy Dodis, Ilya Mironov, and Noah Stephens-Davidowitz. Message transmission with reverse firewalls—secure communication on corrupted machines. In Matthew Robshaw and Jonathan Katz, editors, *Advances in Cryptology – CRYPTO 2016*, pages 341–372, Berlin, Heidelberg, 2016. Springer Berlin Heidelberg. [Cited on page 87]

[DMT19] D. Deuber, B. Magri, and S. A. K. Thyagarajan. Redactable blockchain in the permissionless setting. In *2019 IEEE Symposium on Security and Privacy (SP)*, pages 124–138, 2019. [Cited on page 92]

[Dou02] John R. Douceur. The sybil attack. In Peter Druschel, Frans Kaashoek, and Antony Rowstron, editors, *Peer-to-Peer Systems*, pages 251–260, Berlin, Heidelberg, 2002. Springer Berlin Heidelberg. [Cited on page 65]

[DS15] Jules DuPont and Anna Cinzia Squicciarini. Toward de-anonymizing bitcoin by mapping users location. In *Proceedings of the 5th ACM Conference on Data and Application Security and Privacy*, CODASPY '15, pages 139–141, 2015. [Cited on pages 137 and 138]

[DSG19] Dipankar Dasgupta, John M. Shrein, and Kishor Datta Gupta. A survey of blockchain from security perspective. *Journal of Banking and Financial Technology*, 3(1):1–17, Apr 2019. [Cited on page 135]

[DSLG17] A. Deshpande, K. Stewart, L. Lepetit, and S. Gunashekar. Distributed ledger technologies/blockchain: Challenges, opportunities and the prospects for standards. *British Standards Institution (BSI)*, 2017. Available Online: `https://www.bsigroup.com/LocalFiles/zh-tw/InfoSec-newsletter/No201706/download/BSI_Blockchain_DLT_Web.pdf` (Last accessed 30-May-2020). [Cited on page 1]

[Dwo10] Morris J Dworkin. Recommendation for block cipher modes of operation: Three variants of ciphertext stealing for cbc mode. NIST Pubs, Report Number 800-38A Addendum, 2010. Available Online: `https://www.gpo.gov/fdsys/pkg/GOVPUB-C13-c0b0bae5f66880bf051f6d4ac2d8f07d/pdf/GOVPUB-C13-c0b0bae5f66880bf051f6d4ac2d8f07d.pdf` (Last accessed 08-Feb-2018). [Cited on page 41]

[DY05] Yevgeniy Dodis and Aleksandr Yampolskiy. A verifiable random function with short proofs and keys. In Serge Vaudenay, editor, *Public Key Cryptography - PKC 2005*, pages 416–431, Berlin, Heidelberg, 2005. Springer Berlin Heidelberg. [Cited on page 85]

[ecd14] Simple and secure ECC and ECDSA, 2014. Available Online: `https://github.com/esxgx` (Last accessed 22-March-2020). [Cited on page 56]

[Ell20] Ellipal ltd. Ellipal cold wallet, 2020. Available Online: `https://www.ellipal.com/` (Last accessed 31-May-2020). [Cited on page 14]

[eth20] Ethereum, 2020. Available Online: `https://ethereum.org/` (Last accessed 12-June-2020). [Cited on pages 1 and 3]

[Ett98] J. Mark Ettinger. Steganalysis and game equilibria. In David Aucsmith, editor, *Information Hiding*, pages 319–328, Berlin, Heidelberg, 1998. Springer Berlin Heidelberg. [Cited on page 28]

[Eur16] European Union. Regulation (EU) 2016/679: General Data Protection Regulation, 2016. Available Online: `https://gdpr-info.eu/` (Last accessed 04-May-2020). [Cited on pages 5, 65, and 90]

[FAZ18] D. Frkat, R. Annessi, and T. Zseby. Chainchannels: Private botnet communication over public blockchains. In *2018 IEEE International Conference on Internet of Things (iThings) and IEEE Green Computing and Communications (GreenCom) and IEEE Cyber, Physical and Social Computing (CPSCom) and IEEE Smart Data (SmartData)*, pages 1244–1252, July 2018. [Cited on pages 30 and 78]

[Fel57] William Feller. *An introduction to probability theory and its applications.* A Wiley publication in mathematical statistics. Wiley, New York, 2d ed. edition, 1957. [Cited on page 46]

[FHBS19] M. Florian, S. Henningsen, S. Beaucamp, and B. Scheuermann. Erasing data from blockchain nodes. In *2019 IEEE European Symposium on Security and Privacy Workshops (EuroS PW)*, pages 367–376, June 2019. [Cited on pages 61, 91, 92, and 107]

[FM18] M. Fischlin and S. Mazaheri. Self-guarding cryptographic protocols against algorithm substitution attacks. In *2018 IEEE 31st Computer Security Foundations Symposium (CSF)*, pages 76–90, July 2018. [Cited on pages 27, 88, 89, 90, and 107]

[FN94] Amos Fiat and Moni Naor. Broadcast encryption. In Douglas R. Stinson, editor, *Advances in Cryptology — CRYPTO' 93*, pages 480–491, Berlin, Heidelberg, 1994. Springer Berlin Heidelberg. [Cited on page 69]

[FS07] Eiichiro Fujisaki and Koutarou Suzuki. Traceable ring signature. In Tatsuaki Okamoto and Xiaoyun Wang, editors, *Public Key Cryptography – PKC 2007*, pages 181–200, Berlin, Heidelberg, 2007. Springer Berlin Heidelberg. [Cited on page 19]

[Fuj11] Eiichiro Fujisaki. Sub-linear size traceable ring signatures without random oracles. In Aggelos Kiayias, editor, *Topics in Cryptology – CT-RSA 2011*, pages 393–415, Berlin, Heidelberg, 2011. Springer Berlin Heidelberg. [Cited on page 19]

[FV17a] G. Fanti and P. Viswanath. Anonymity properties of the bitcoin P2P network. *CoRR*, abs/1703.08761, 2017. [Cited on pages 2, 139, and 145]

[FV17b] Giulia Fanti and Pramod Viswanath. Deanonymization in the bitcoin p2p network. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems 30*, pages 1364–1373. Curran Associates, Inc., 2017. [Cited on page 2]

[FWB15] Martin Florian, Johannes Walter, and Ingmar Baumgart. Sybil-resistant pseudonymization and pseudonym change without trusted third parties. In *Proceedings of the 14th ACM Workshop on Privacy in the Electronic Society*, WPES '15, page 65–74, New York, NY, USA, 2015. Association for Computing Machinery. [Cited on page 61]

[GBPG03] Eu-Jin Goh, Dan Boneh, Benny Pinkas, and Philippe Golle. The design and implementation of protocol-based hidden key recovery. In Colin Boyd and Wenbo Mao, editors, *Information Security*, pages 165–179, Berlin, Heidelberg, 2003. Springer Berlin Heidelberg. [Cited on pages 25, 32, 37, 38, and 47]

[Giz17] Giza Device Ltd. Giza wallet, 2017. Avialable Online: `https://www.gizadevice.com/` (Last accessed 7-Feb-2018). [Cited on page 38]

[GKRN18] Steven Goldfeder, Harry Kalodner, Dillon Reisman, and Arvind Narayanan. When the cookie meets the blockchain: Privacy risks of web payments via cryptocurrencies. *Proceedings on Privacy Enhancing Technologies*, 2018(4):179 – 199, 2018. [Cited on page 139]

[GKZ06] Zbigniew Gołebiewski, Mirosław Kutyłowski, and Filip Zagórski. Stealing secrets with ssl/tls and ssh – kleptographic attacks. In David Pointcheval, Yi Mu, and Kefei Chen, editors, *Cryptology and Network Security*, pages 191–202, Berlin, Heidelberg, 2006. Springer Berlin Heidelberg. [Cited on page 25]

[GMR88] Shafi Goldwasser, Silvio Micali, and Ronald L. Rivest. A digital signature scheme secure against adaptive chosen-message attacks. *SIAM Journal on Computing*, 17(2):281–308, 1988. [Cited on page 17]

[Gün12] Oliver Günther. Broadcast key encapsulation mechanism github repository, 2012. Available Online: `https://github.com/oliverguenther/PBC_BKEM` (Last accessed 08-May-2019). [Cited on page 70]

[Gop03] K. Gopalan. Audio steganography using bit modification. In *2003 International Conference on Multimedia and Expo. ICME '03. Proceedings (Cat. No.03TH8698)*, volume 1, pages I–629, 2003. [Cited on page 29]

[Gur18] M. Guri. Beatcoin: Leaking private keys from air-gapped cryptocurrency wallets. In *2018 IEEE International Conference on Internet of Things (iThings) and IEEE Green Computing and Communications (GreenCom) and IEEE Cyber, Physical and Social Computing (CPSCom) and IEEE Smart Data (SmartData)*, pages 1308–1316, 2018. [Cited on pages 14 and 15]

[H. 18] H. Shaban. People are using bitcoin's system to share child pornography, researchers say, March 2018. Available Online: `https://www.washingtonpost.com/news/the-switch/wp/2018/03/22/people-are-using-bitcoins-system-to-share-child-pornography/` (Last accessed 17-March-2020). [Cited on pages 5 and 61]

[HAZ17] Alexander Hartl, Robert Annessi, and Tanja Zseby. A subliminal channel in eddsa: Information leakage with high-speed signatures. In *Proceedings of the 2017 International Workshop on Managing Insider Security Threats*, MIST '17, page 67–78, New York, NY, USA, 2017. Association for Computing Machinery. [Cited on pages 7, 26, 29, and 87]

[HBHW18] Daira Hopwood, Sean Bowe, Taylor Hornby, and Nathan Wilcox. Zcash protocol specification, 2018. Available Online: `https://cryptoverze-0m5mfism.netdna-ssl.com/wp-content/uploads/2019/01/z-cash-zec-whitepaper.pdf` (Last accessed 22-Aug-2019). [Cited on pages 136 and 142]

[HKK17] L. Hanzlik, K. Kluczniak, and M. Kutyłowski. Controlled randomness – a defense against backdoors in cryptographic devices. In Raphaël C.-W. Phan and Moti Yung, editors, *Paradigms in Cryptology – Mycrypt 2016. Malicious and Exploratory Cryptology*, pages 215–232, Cham, 2017. Springer International Publishing. [Cited on pages 26, 85, and 107]

[HLvA02] Nicholas J. Hopper, John Langford, and Luis von Ahn. Provably secure steganography. In *CRYPTO 2002*, 2002. [Cited on page 28]

[HvL09] N. Hopper, L. von Ahn, and J. Langford. Provably secure steganography. *IEEE Transactions on Computers*, 58(5):662–676, 2009. [Cited on pages 28, 31, 58, 60, and 61]

[J.D17] J.D. Bruce. The Mini-Blockchain Scheme - Rev. 3, 2017. Available Online: `http://cryptonite.info/files/mbc-scheme-rev3.pdf` (Last accessed 08-November-2018). [Cited on pages 91 and 107]

[JHW18] A. P. Joshi, M. Han, and Y. Wang. A survey on security and privacy issues of blockchain technology. *Mathematical Foundations of Computing*, 2018. [Cited on page 135]

[JSKV18] P. L. Juhász, J. Stéger, D. Kondor, and G. Vattay. A bayesian approach to identify bitcoin users. *PLOS ONE*, 13(12):1–21, 12 2018. [Cited on page 138]

[KHD17] K. Korpela, J. Hallikas, and T. Dahlberg. Digital Supply Chain Transformation toward Blockchain Integration. In *Proceedings of the 50th Hawaii International Conference on System Sciences*, 2017. [Cited on page 1]

[KKM14] Philip Koshy, Diana Koshy, and Patrick McDaniel. An analysis of anonymity in bitcoin using p2p network traffic. In Nicolas Christin and Reihaneh Safavi-Naini, editors, *Financial Cryptography and Data Security*, pages 469–485, Berlin, Heidelberg, 2014. Springer Berlin Heidelberg. [Cited on pages 104, 137, 138, and 145]

[KL18] M. C. Kus Khalilov and A. Levi. A survey on anonymity and privacy in bitcoin-like digital cash systems. *IEEE Communications Surveys Tutorials*, 20(3):2543–2585, 2018. [Cited on page 135]

[KP02] Stefan Katzenbeisser and Fabien A. P. Petitcolas. Defining security in steganographic systems. In Edward J. Delp III and Ping Wah Wong, editors, *Security and Watermarking of Multimedia Contents IV*, volume 4675, pages 50 – 56. International Society for Optics and Photonics, SPIE, 2002. [Cited on pages 29 and 77]

[KR00] Hugo Krawczyk and Tal Rabin. Chameleon signatures. In *NDSS*, 2000. [Cited on page 92]

[KRC18] Jeffrey Knockel, Thomas Ristenpart, and Jedidiah R. Crandall. When textbook RSA is used to protect the privacy of hundreds of millions of users. *CoRR*, abs/1802.03367, 2018. [Cited on page 36]

[KV09] Jonathan Katz and Vinod Vaikuntanathan. Signature schemes with bounded leakage resilience. In *ASIACRYPT 2009*, 2009. [Cited on page 35]

[Lab14] Kaspersky Lab. Kaspersky Lab Report: Financial cyber threats in 2013. Part 2: malware, April 2014. Available Online: `https://securelist.com/financial-cyber-threats-in-2013-part-2-malware/59414/` (Last accessed 21-June-2020). [Cited on pages 38 and 48]

[LCO+16] Loi Luu, Duc-Hiep Chu, Hrishi Olickel, Prateek Saxena, and Aquinas Hobor. Making smart contracts smarter. In *CCS 2016*, pages 254–269, 2016. [Cited on page 36]

[Led20] Ledger SAS. Ledger wallet, 2020. Available Online: `https://www.ledger.com/` (Last accessed 31-May-2020). [Cited on page 14]

[LJC+17] Xiaoqi Li, Peng Jiang, Ting Chen, Xiapu Luo, and Qiaoyan Wen. A survey on the security of blockchain systems. *Future Generation Computer Systems*, 2017. [Cited on page 135]

[LR86] Michael Luby and Charles Rackoff. How to construct pseudo-random permutations from pseudo-random functions. In Hugh C. Williams, editor,

Advances in Cryptology — CRYPTO '85 Proceedings, pages 447–447, Berlin, Heidelberg, 1986. Springer Berlin Heidelberg. [Cited on page 15]

[LSP82] L. LAMPORT, R. SHOSTAK, and M. PEASE. The Byzantine Generals Problem. *ACM Transactions on Programming Languages and Systems*, 4(3):382–401, July 1982. [Cited on page 14]

[LTKS15] Loi Luu, Jason Teutsch, Raghav Kulkarni, and Prateek Saxena. Demystifying incentives in the consensus computer. In *Proceedings of the 22Nd ACM SIGSAC Conference on Computer and Communications Security*, CCS '15, pages 706–719, New York, NY, USA, 2015. ACM. [Cited on page 105]

[lUJ17] Svein Ølnes, Jolien Ubacht, and Marijn Janssen. Blockchain in government: Benefits and implications of distributed ledger technology for information sharing. *Government Information Quarterly*, 34(3):355 – 364, 2017. [Cited on pages 1 and 14]

[LW05] Joseph K. Liu and Duncan S. Wong. Linkable ring signatures: Security models and new schemes. In Osvaldo Gervasi, Marina L. Gavrilova, Vipin Kumar, Antonio Laganà, Heow Pueh Lee, Youngsong Mun, David Taniar, and Chih Jeng Kenneth Tan, editors, *Computational Science and Its Applications – ICCSA 2005*, pages 614–623, Berlin, Heidelberg, 2005. Springer Berlin Heidelberg. [Cited on page 19]

[LWW04] Joseph K. Liu, Victor K. Wei, and Duncan S. Wong. Linkable spontaneous anonymous group signature for ad hoc groups. In Huaxiong Wang, Josef Pieprzyk, and Vijay Varadharajan, editors, *Information Security and Privacy*, pages 325–335, Berlin, Heidelberg, 2004. Springer Berlin Heidelberg. [Cited on page 19]

[mar19] Silk Road (marketplace), 2019. Available Online: `https://en.wikipedia.org/wiki/Silk_Road_(marketplace)` (Last accessed 22-Aug-2019). [Cited on page 138]

[Max] Greg Maxwell. Confidential transactions. Available Online: `https://people.xiph.org/~greg/confidential_values.txt` (Last accessed 25-June-2019). [Cited on pages 53 and 136]

[Max13a] G. Maxwell. Coinjoin: Bitcoin privacy for the real world, 8 2013. Available Online : `https://bitcointalk.org/index.php?topic=279249.0` (Last accessed 25-Oct.-2018). [Cited on page 141]

[Max13b] Greg Maxwell. Coinswap: Transaction graph disjoint trustless trading, 2013. Available Online: `https://bitcointalk.org/index.php?topic=321228.0` (Last accessed 12-Aug-2019). [Cited on page 141]

[MBR99] L. M. Marvel, C. G. Boncelet, and C. T. Retter. Spread spectrum image steganography. *IEEE Transactions on Image Processing*, 8(8):1075–1083, 1999. [Cited on page 29]

[MCAF20] D.J. Munoz, D.A. Constantinescu, R. Asenjo, and L. Fuentes. ClinicAppChain: A Low-Cost Blockchain Hyperledger Solution for Healthcare. In Javier Prieto, Ashok Kumar Das, Stefano Ferretti, António Pinto, and Juan Manuel Corchado, editors, *Blockchain and Applications*, pages 36–44, Cham, 2020. Springer International Publishing. [Cited on page 1]

[ME10] Elsayed Mohamed and Hassan Elkamchouchi. Kleptographic Attacks on Elliptic Curve Cryptosystems. *Journal of Computer Science*, 10(6):213–215, 2010. [Cited on page 44]

[Med20] Medicalchain. Medicalchain whitepaper 2.1, 2020. Available online: `https://medicalchain.com/Medicalchain-Whitepaper-EN.pdf`, (Last accessed 30-May-2020). [Cited on page 1]

[MGGR13] I. Miers, C. Garman, M. Green, and A. D. Rubin. Zerocoin: Anonymous distributed e-cash from bitcoin. In *2013 IEEE Symposium on Security and Privacy*, pages 397–411, 5 2013. [Cited on pages 1, 136, 141, and 143]

[MHH+16] Roman Matzutt, Oliver Hohlfeld, Martin Henze, Robin Rawiel, Jan Henrik Ziegeldorf, and Klaus Wehrle. Poster: I don't want that content! on the risks of exploiting bitcoin's blockchain as a content store. In *CCS '16*, 2016. [Cited on pages 5, 29, 30, 31, 61, 65, and 77]

[MHH+18] Roman Matzutt, Jens Hiller, Martin Henze, Jan Henrik Ziegeldorf, Dirk Müllmann, Oliver Hohlfeld, and Klaus Wehrle. A quantitative analysis of the impact of arbitrary blockchain content on bitcoin. In Sarah Meiklejohn and Kazue Sako, editors, *Financial Cryptography and Data Security*, pages 420–438, Berlin, Heidelberg, 2018. Springer Berlin Heidelberg. [Cited on pages 5, 30, 61, 65, and 77]

[MHZ+18] R. Matzutt, M. Henze, J. H. Ziegeldorf, J. Hiller, and K. Wehrle. Thwarting unwanted blockchain content insertion. In *IC2E 2018*, pages 364–370, April 2018. [Cited on pages 6, 31, 62, 92, 93, and 107]

[Min18] Mini-blockchain Project. Cryptonite Cryptocurrency, 2018. Available Online: `http://cryptonite.info/` (Last accessed 01-November-2018). [Cited on pages 31 and 91]

[Mit00] Thomas Mittelholzer. An information-theoretic approach to steganography and watermarking. In Andreas Pfitzmann, editor, *Information Hiding*, pages 1–16, Berlin, Heidelberg, 2000. Springer Berlin Heidelberg. [Cited on pages 27, 28, and 29]

[MLC01] Ira S. Moskowitz, Garth E. Longdon, and LiWu Chang. A new paradigm hidden in steganography. In *Proceedings of the 2000 Workshop on New Security Paradigms*, NSPW '00, page 41–50, New York, NY, USA, 2001. Association for Computing Machinery. [Cited on page 29]

[MLS+15] Emily McReynolds, Adam Lerner, Will Scott, Franziska Roesner, and Tadayoshi Kohno. Cryptographic currencies from a tech-policy perspective: Policy issues and technical directions. In Michael Brenner, Nicolas Christin,

Benjamin Johnson, and Kurt Rohloff, editors, *Financial Cryptography and Data Security*, pages 94–111, Berlin, Heidelberg, 2015. Springer Berlin Heidelberg. [Cited on pages 13 and 14]

[MMSK18] M. Minaei, P. Moreno-Sanchez, and A. Kate. R3c3: Cryptographically secure censorship resistant rendezvous using cryptocurrencies. Cryptology ePrint Archive, Report 2018/454, 2018. Available online: `https://eprint.iacr.org/2018/454` (Last accessed 14-Feb-2019). [Cited on pages 5, 30, 61, 76, and 77]

[MNB+17] Esther Mengelkamp, Benedikt Notheisen, Carolin Beer, David Dauer, and Christof Weinhardt. A blockchain-based smart grid: towards sustainable local energy markets. *Computer Science - Research and Development*, pages 1–8, 08 2017. [Cited on page 1]

[Mon18] Monero. Monero, 2018. Available Online: `https://getmonero.org/` (Last accessed 07-Feb-2018). [Cited on pages 8 and 10]

[mon19] Monero Project, 2019. Available Online: `https://github.com/monero-project/monero` (Last accessed 16-Aug-2019). [Cited on pages 136 and 144]

[MP15] Gregory Maxwell and Andrew Poelstra. Borromean Ring Signatures, 2015. Available Online: `http://diyhpl.us/~bryan/papers2/bitcoin/Borromean%20ring%20signatures.pdf` (Last accessed 07-Feb-2018). [Cited on pages 21, 53, 54, and 143]

[MP18] Indra Deep Mastan and Souradyuti Paul. A new approach to deanonymization of unreachable bitcoin nodes. In Srdjan Capkun and Sherman S. M. Chow, editors, *Cryptology and Network Security*, pages 277–298, Cham, 2018. Springer International Publishing. [Cited on page 139]

[MPJ+13] Sarah Meiklejohn, Marjori Pomarole, Grant Jordan, Kirill Levchenko, Damon McCoy, Geoffrey M. Voelker, and Stefan Savage. A fistful of bitcoins: Characterizing payments among men with no names. In *Proceedings of the 2013 Conference on Internet Measurement Conference*, IMC '13, pages 127–140, New York, NY, USA, 2013. ACM. [Cited on pages 137 and 138]

[MRV99] S. Micali, M. Rabin, and S. Vadhan. Verifiable random functions. In *40th Annual Symposium on Foundations of Computer Science*, pages 120–130, Oct 1999. [Cited on pages 26, 85, and 107]

[MS17] Albert Molina and Herman Schoenfeld. PascalCoin Whitepaper v2, 2017. Available Online: `https://www.pascalcoin.org/PascalCoinWhitePaperV2.pdf` (Last accessed 08-November-2018). [Cited on pages 31, 90, 93, and 107]

[MSD15] Ilya Mironov and Noah Stephens-Davidowitz. Cryptographic reverse firewalls. In Elisabeth Oswald and Marc Fischlin, editors, *Advances in Cryptology - EUROCRYPT 2015*, pages 657–686, Berlin, Heidelberg, 2015. Springer Berlin Heidelberg. [Cited on pages 25 and 87]

[MSH+18] Malte Möser, Kyle Soska, Ethan Heilman, Kevin Lee, Henry Heffan, Shashvat Srivastava, Kyle Hogan, Jason Hennessey, Andrew Miller, Arvind Narayanan, and Nicolas Christin. An empirical analysis of traceability in the monero blockchain. *Proceedings on Privacy Enhancing Technologies*, 2018(3), 2018. [Cited on pages 139 and 140]

[Nak08] Satoshi Nakamoto. A Peer-to-Peer Electronic Cash System, 2008. Available Online: `https://bitcoin.org/bitcoin.pdf` (Last accessed 05-Nov-2018). [Cited on pages 1, 3, 13, 14, 137, and 138]

[NBF+16] Arvind Narayanan, Joseph Bonneau, Edward Felten, Andrew Miller, and Steven Goldfeder. *Bitcoin and cryptocurrency technologies: a comprehensive introduction*. Princeton University Press, 2016. [Cited on pages 2, 6, and 14]

[Ngu04] Phong Q. Nguyen. Can we trust cryptographic software? cryptographic flaws in gnu privacy guard v1.2.3. In Christian Cachin and Jan L. Camenisch, editors, *Advances in Cryptology - EUROCRYPT 2004*, pages 555–570, Berlin, Heidelberg, 2004. Springer Berlin Heidelberg. [Cited on page 36]

[NNM14] Surae. Noether, Sarang Noether, and A. Mackenzie. A Note on Chain Reactions in Traceability in CryptoNote 2.0, 9 2014. Available Online: `https://ww.getmonero.org/resources/research-lab/pubs/MRL-0001.pdf` (Last accessed 01-Jan.-2019). [Cited on page 140]

[O'L19] Rachel Rose O'Leary. Monero Fees Fall to Almost Zero After 'Bulletproofs' Upgrade, 2019. Available Online: `https://www.coindesk.com/monero-fees-fall-to-almost-zero-after-bulletproofs-upgrade` (Last accessed 11-February-2020). [Cited on page 145]

[OME98] J. A. O'Sullivan, P. Moulin, and J. M. Ettinger. Information theoretic analysis of steganography. In *Proceedings. 1998 IEEE International Symposium on Information Theory*, 1998. [Cited on page 27]

[PAK99] F. A. P. Petitcolas, R. J. Anderson, and M. G. Kuhn. Information hiding-a survey. *Proceedings of the IEEE*, 87(7):1062–1078, 1999. [Cited on pages 29 and 59]

[Par18] Juha Partala. Provably secure covert communication on blockchain. *Cryptography*, 2(3), 2018. [Cited on pages 31, 63, and 64]

[PCAP19] E. Politou, F. Casino, E. Alepis, and C. Patsakis. Blockchain mutability: Challenges and proposed solutions. *IEEE Transactions on Emerging Topics in Computing*, pages 1–1, 2019. [Cited on pages 2, 65, 81, 90, and 92]

[PDC17] Ivan Puddu, Alexandra Dmitrienko, and Srdjan Capkun. μchain: How to forget without hard forks. Cryptology ePrint Archive, Report 2017/106, 2017. `https://eprint.iacr.org/2017/106`. [Cited on pages 5, 62, 65, 92, and 107]

[Ped92] Torben Pryds Pedersen. Non-interactive and information-theoretic secure verifiable secret sharing. In Joan Feigenbaum, editor, *Advances in Cryptology*

— *CRYPTO '91*, pages 129–140, Berlin, Heidelberg, 1992. Springer Berlin Heidelberg. [Cited on page 142]

[Poe18] Andrew Poelstra. Bulletproofs: Faster Rangeproofs and Much More, 2018. Available Online: `https://blockstream.com/2018/02/21/en-bulletproofs-faster-rangeproofs-and-much-more/` (Last accessed 13-Aug-2019). [Cited on page 140]

[Por13] T. Pornin. Deterministic DSA and ECDSA. RFC 6979, RFC Editor, August 2013. Available online: `https://tools.ietf.org/html/rfc6979` (Last accessed 13-Feb-2019). [Cited on page 84]

[PP10] Christof Paar and Jan Pelzl. *Understanding cryptography : a textbook for students and practitioners*. Springer, Berlin, 2010. [Cited on pages 23 and 36]

[PS96] David Pointcheval and Jacques Stern. Security proofs for signature schemes. In Ueli Maurer, editor, *Advances in Cryptology — EUROCRYPT '96*, pages 387–398, Berlin, Heidelberg, 1996. Springer Berlin Heidelberg. [Cited on page 17]

[PS16] David Pointcheval and Olivier Sanders. Short randomizable signatures. In Kazue Sako, editor, *Topics in Cryptology - CT-RSA 2016*, pages 111–126, Cham, 2016. Springer International Publishing. [Cited on pages 96, 102, 103, 104, 107, and 154]

[Qua11] QuantumMechanic. Proof of stake instead of proof of work, july 2011. Available Online: `https://bitcointalk.org/index.php?topic=27787.0` (Last accessed 30-May-2020). [Cited on page 2]

[RC19] Ruben Recabarren and Bogdan Carbunar. Tithonus: A bitcoin based censorship resilient system. *Proceedings on Privacy Enhancing Technologies*, 2019(1):68 – 86, 2019. [Cited on pages 5, 30, 61, 76, 77, and 78]

[rep20] Reproducible builds, 2020. Available Online: `https://reproducible-builds.org/` (Last accessed 20-May-2020). [Cited on page 83]

[RH11] F. Reid and M. Harrigan. An analysis of anonymity in the bitcoin system. In *2011 IEEE Third International Conference on Privacy, Security, Risk and Trust and 2011 IEEE Third International Conference on Social Computing*, pages 1318–1326, 10 2011. [Cited on pages 137 and 138]

[RMSK14] Tim Ruffing, Pedro Moreno-Sanchez, and Aniket Kate. Coinshuffle: Practical decentralized coin mixing for bitcoin. In Mirosław Kutyłowski and Jaideep Vaidya, editors, *Computer Security - ESORICS 2014*, pages 345–364, Cham, 2014. Springer International Publishing. [Cited on page 141]

[RP97] Vincent Rijmen and Bart Preneel. A family of trapdoor ciphers. In Eli Biham, editor, *Fast Software Encryption*, pages 139–148, Berlin, Heidelberg, 1997. Springer Berlin Heidelberg. [Cited on page 26]

[RS13] Dorit Ron and Adi Shamir. Quantitative analysis of the full bitcoin transaction graph. In Ahmad-Reza Sadeghi, editor, *Financial Cryptography and Data Security*, pages 6–24, Berlin, Heidelberg, 2013. Springer Berlin Heidelberg. [Cited on pages 137 and 138]

[RS14] Dorit Ron and Adi Shamir. How did dread pirate roberts acquire and protect his bitcoin wealth? In Rainer Böhme, Michael Brenner, Tyler Moore, and Matthew Smith, editors, *Financial Cryptography and Data Security*, pages 3–15, Berlin, Heidelberg, 2014. Springer Berlin Heidelberg. [Cited on pages 137 and 138]

[RST01] Ronald L. Rivest, Adi Shamir, and Yael Tauman. How to leak a secret. In Colin Boyd, editor, *Advances in Cryptology — ASIACRYPT 2001*, pages 552–565, Berlin, Heidelberg, 2001. Springer Berlin Heidelberg. [Cited on pages 2, 8, 18, 19, 39, 85, 97, and 135]

[RTYZ16a] Alexander Russell, Qiang Tang, Moti Yung, and Hong-Sheng Zhou. Cliptography: Clipping the power of kleptographic attacks. In *ASIACRYPT*, 2016. [Cited on pages 27, 86, and 107]

[RTYZ16b] Alexander Russell, Qiang Tang, Moti Yung, and Hong-Sheng Zhou. Destroying steganography via amalgamation: Kleptographically cpa secure public key encryption. Cryptology ePrint Archive, Report 2016/530, 2016. [Cited on pages 27, 62, 86, and 107]

[RTYZ17] Alexander Russell, Qiang Tang, Moti Yung, and Hong-Sheng Zhou. Generic semantic security against a kleptographic adversary. CCS '17, pages 907–922, New York, NY, USA, 2017. ACM. [Cited on pages 27, 86, and 107]

[S. 18] S. Gibbs. Child abuse imagery found within bitcoin's blockchain, March 2018. Available Online: `https://www.theguardian.com/technology/2018/mar/20/child-abuse-imagery-bitcoin-blockchain-illegal-content` (Last accessed 17-March-2020). [Cited on pages 5 and 61]

[Sab13] Nicolas Van Saberhagen. Cryptonote v 2.0, 2013. whitepaper, Available online: `https://cryptonote.org/whitepaper.pdf`, (Last accessed 23-May-2019). [Cited on pages 19, 39, 49, 97, and 144]

[Sat20] SatoshiLabs. Trezor wallet, 2020. [Cited on page 14]

[SCG+14] E. B. Sasson, A. Chiesa, C. Garman, M. Green, I. Miers, E. Tromer, and M. Virza. Zerocash: Decentralized anonymous payments from bitcoin. In *2014 IEEE Symposium on Security and Privacy*, pages 459–474, 5 2014. [Cited on page 143]

[Sch91] C. P. Schnorr. Efficient signature generation by smart cards. *Journal of Cryptology*, 4:161–174, January 1991. [Cited on pages 27 and 87]

[Sch19] Bruce Schneier. There's no good reason to trust blockchain technology, 2019. Available Online: `https://www.wired.com/story/theres-no-good-reason-to-trust-blockchain-technology/?verso=true` (Last accessed 02-August-2019). [Cited on page 5]

[SFKR15] Bruce Schneier, Matthew Fredrikson, Tadayoshi Kohno, and Thomas Ristenpart. Surreptitiously weakening cryptographic systems. Cryptology ePrint Archive, Report 2015/097, 2015. Available Online: `https://eprint.iacr.org/2015/097` (Last accessed 21-March-2020). [Cited on pages 26, 36, and 37]

[Shi14] Ken Shirriff. Hidden surprises in the bitcoin blockchain and how they are stored: Nelson mandela, wikileaks, photos, and python software, 2014. Available Online: `http://www.righto.com/2014/02/ascii-bernanke-wikileaks-photographs.html` (Last accessed 01-November-2018). [Cited on page 31]

[Sie16] David Siegel. Understanding the dao attack, 2016. Available Online: `https://www.coindesk.com/understanding-dao-hack-journalists` (Last accessed 7-Feb-2018). [Cited on page 37]

[Sim84] Gustavus J. Simmons. *The Prisoners' Problem and the Subliminal Channel*, pages 51–67. Springer US, Boston, MA, 1984. [Cited on pages 3, 6, 10, 27, and 58]

[Sky18] Sky News. Child abuse images hidden in bitcoin blockchain, March 2018. Available Online: `https://news.sky.com/story/child-abuse-images-hidden-in-bitcoin-blockchain-11298022` (Last accessed 17-March-2020). [Cited on pages 5 and 61]

[SMZ14] Michele Spagnuolo, Federico Maggi, and Stefano Zanero. Bitiodine: Extracting intelligence from the bitcoin network. In Nicolas Christin and Reihaneh Safavi-Naini, editors, *Financial Cryptography and Data Security*, pages 457–468, Berlin, Heidelberg, 2014. Springer Berlin Heidelberg. [Cited on page 139]

[STW+16] James Smith, Jeni Tennison, Peter Wells, Jamie Fawcett, and Stuart Harrison. Applying blockchain technology in global data infrastructure. Technical report, Open Data Institute, June 2016. ODI-TR-2016-001. [Cited on page 61]

[SVS18] Andrew Sward, Ivy Vecna, and Forrest Stonedahl. Data insertion in bitcoin's blockchain. *Ledger*, 3, Apr. 2018. [Cited on page 30]

[Tap17] D. Tapscott, A.and Tapscott. How Blockchain Is Changing Finance, March 2017. Available Online: `https://hbr.org/2017/03/how-blockchain-is-changing-finance` (Last accessed 12-June-2020). [Cited on page 1]

[TD17] A. Tomescu and S. Devadas. Catena: Efficient non-equivocation via bitcoin. In *2017 IEEE Symposium on Security and Privacy (SP)*, volume 00, pages 393–409, May 2017. [Cited on pages 5, 30, 61, and 78]

[Tea18] Bytecoin Developers Team. Bytecoin project github repository, 2018. Available Online: `https://github.com/bcndev` (Last accessed 26-Nov-2018). [Cited on pages 49 and 71]

[Teş19] George Teşeleanu. Threshold kleptographic attacks on discrete logarithm based signatures. In Tanja Lange and Orr Dunkelman, editors, *Progress in Cryptology – LATINCRYPT 2017*, pages 401–414, Cham, 2019. Springer International Publishing. [Cited on pages 26 and 34]

[TLK+18] Muoi Tran, Loi Luu, Min Suk Kang, Iddo Bentov, and Prateek Saxena. Obscuro: A bitcoin mixer using trusted execution environments. In *Proceedings of the 34th Annual Computer Security Applications Conference*, ACSAC '18, pages 692–701, New York, NY, USA, 2018. ACM. [Cited on page 141]

[tor19] Tor, 2019. Available Online: `https://www.torproject.org/` (Last accessed 10-Aug-2019). [Cited on page 139]

[TS16] F. Tschorsch and B. Scheuermann. Bitcoin and beyond: A technical survey on decentralized digital currencies. *IEEE Communications Surveys Tutorials*, 18(3):2084–2123, 2016. [Cited on pages 137 and 138]

[VR15] Luke Valenta and Brendan Rowan. Blindcoin: Blinded, accountable mixes for bitcoin. In Michael Brenner, Nicolas Christin, Benjamin Johnson, and Kurt Rohloff, editors, *Financial Cryptography and Data Security*, pages 112–126, Berlin, Heidelberg, 2015. Springer Berlin Heidelberg. [Cited on page 141]

[WBDY98] Hongjun Wu, Feng Bao, Robert H. Deng, and Qin Zhong Ye. Cryptanalysis of rijmen-preneel trapdoor ciphers. In Kazuo Ohta and Dingyi Pei, editors, *Advances in Cryptology — ASIACRYPT'98*, pages 126–132, Berlin, Heidelberg, 1998. Springer Berlin Heidelberg. [Cited on page 26]

[Wei03] Eric W. Weisstein. RSA-576 Factored, 2003. Available Online: `https://mathworld.wolfram.com/news/2003-12-05/rsa/` (Last accessed 16-April-2020). [Cited on page 24]

[Wik18] Bitcoin Wiki. Technical background of version 1 bitcoin addresses, 2018. Available Online : `https://en.bitcoin.it/wiki/Technical_background_of_version_1_Bitcoin_addresses`(Last accessed 15-Oct.-2018). [Cited on page 137]

[WLS+18] D.A. Wijaya, J. Liu, R. Steinfeld, D. Liu, and T.H. Yuen. Anonymity reduction attacks to monero, 2018. Available Online: `http://xxhb.fjnu.edu.cn/_upload/tpl/06/5d/1629/template1629/papers/59.pdf` (Last accessed 01-Jan.-2019). [Cited on page 140]

[WLS+19] Dimaz Ankaa Wijaya, Joseph K. Liu, Ron Steinfeld, Dongxi Liu, and Jiangshan Yu. On the unforkability of monero. In *Proceedings of the 2019 ACM Asia Conference on Computer and Communications Security*, Asia CCS '19, pages 621–632, New York, NY, USA, 2019. ACM. [Cited on pages 140 and 145]

[WLSL18] D. A. Wijaya, J. Liu, R. Steinfeld, and D. Liu. Monero ring attack: Recreating zero mixin transaction effect. In *2018 17th IEEE International Conference On Trust, Security And Privacy In Computing And Communications/ 12th IEEE International Conference On Big Data Science And Engineering (TrustCom/BigDataSE)*, pages 1196–1201, 8 2018. [Cited on page 140]

[Wui15] P. Wuille. Bitcoin commit 5400ef6, 2015. Available Online: `https://github.com/bitcoin/bitcoin/commit/5400ef6bcb9d243b2b21697775aa6491115420f3` (Last accessed 28-Jan-2019). [Cited on page 139]

[YY96] Adam Young and Moti Yung. The dark side of "black-box" cryptography or: Should we trust capstone? In *CRYPTO '96*, 1996. [Cited on pages 4, 6, 22, 24, 26, 32, 82, 83, 84, 107, and 110]

[YY97a] Adam Young and Moti Yung. Kleptography: Using cryptography against cryptography. In *EUROCRYPT '97*, 1997. [Cited on pages 4, 6, 10, 22, 24, 32, 84, and 110]

[YY97b] Adam Young and Moti Yung. The prevalence of kleptographic attacks on discrete-log based cryptosystems. In *CRYPTO '97*, 1997. [Cited on pages 4, 22, 23, 34, and 110]

[YY98] Adam Young and Moti Yung. Monkey: Black-Box Symmetric Ciphers Designed for MONopolizing KEYs. In Serge Vaudenay, editor, *Fast Software Encryption*, pages 122–133, Berlin, Heidelberg, 1998. Springer Berlin Heidelberg. [Cited on pages 22 and 23]

[YY03] Adam Young and Moti Yung. Backdoor attacks on black-box ciphers exploiting low-entropy plaintexts. In Rei Safavi-Naini and Jennifer Seberry, editors, *Information Security and Privacy*, pages 297–311, Berlin, Heidelberg, 2003. Springer Berlin Heidelberg. [Cited on pages 22 and 23]

[YY04] Adam Young and Moti Yung. *Malicious cryptography exposing cryptovirology*. Wiley, Hoboken, N.J., 2004. [Cited on pages 4 and 22]

[YY05a] Adam Young and Moti Yung. Malicious cryptography: Kleptographic aspects. In Alfred Menezes, editor, *Topics in Cryptology – CT-RSA 2005*, pages 7–18, Berlin, Heidelberg, 2005. Springer Berlin Heidelberg. [Cited on pages 22, 24, and 36]

[YY05b] Adam Young and Moti Yung. A subliminal channel in secret block ciphers. In Helena Handschuh and M. Anwar Hasan, editors, *SAC'04*, pages 198–211, Berlin, Heidelberg, 2005. Springer. [Cited on page 22]

[YY05c] Adam Young and Moti Yung. A subliminal channel in secret block ciphers. In Helena Handschuh and M. Anwar Hasan, editors, *Selected Areas in Cryptography*, pages 198–211, Berlin, Heidelberg, 2005. Springer Berlin Heidelberg. [Cited on pages 22 and 23]

[YY06] Adam Young and Moti Yung. A space efficient backdoor in rsa and its applications. In Bart Preneel and Stafford Tavares, editors, *Selected Areas in Cryptography*, pages 128–143, Berlin, Heidelberg, 2006. Springer Berlin Heidelberg. [Cited on pages 22 and 24]

[YY07] Adam L. Young and Moti M. Yung. Space-efficient kleptography without random oracles. In Teddy Furon, François Cayre, Gwenaël Doërr, and Patrick

Bas, editors, *Information Hiding*, pages 112–129, Berlin, Heidelberg, 2007. Springer Berlin Heidelberg. [Cited on pages 22 and 24]

[YY10] Adam Young and Moti Yung. Kleptography from standard assumptions and applications. In Juan A. Garay and Roberto De Prisco, editors, *Security and Cryptography for Networks*, pages 271–290, Berlin, Heidelberg, 2010. Springer Berlin Heidelberg. [Cited on pages 22 and 24]

[ZBA18] Dirk A. Zetzsche, Ross P. Buckley, and Douglas W. Arner. The distributed liability of distributed ledgers: Legal risks of blockchain. *University of Illinois Law Review*, pages 1361–1406, 2018. [Cited on page 65]

[zca20] Zcash, 2020. Available Online: `https://z.cash` (Last accessed 21-July-2020). [Cited on pages 5 and 143]

[ZFK+98] J. Zöllner, H. Federrath, H. Klimant, A. Pfitzmann, R. Piotraschke, A. Westfeld, G. Wicke, and G. Wolf. Modeling the security of steganographic systems. In *Information Hiding*, 1998. [Cited on pages 28 and 29]

[ZGH+15] Jan Henrik Ziegeldorf, Fred Grossmann, Martin Henze, Nicolas Inden, and Klaus Wehrle. Coinparty: Secure multi-party mixing of bitcoins. In *Proceedings of the 5th ACM Conference on Data and Application Security and Privacy*, CODASPY '15, pages 75–86, New York, NY, USA, 2015. ACM. [Cited on page 141]

[ZLLZ13] Yinghui Zhang, Hui Li, Xiaoqing Li, and Hui Zhu. Provably secure and subliminal-free variant of schnorr signature. In Khabib Mustofa, Erich J. Neuhold, A. Min Tjoa, Edgar Weippl, and Ilsun You, editors, *Information and Communication Technology*, pages 383–391, Berlin, Heidelberg, 2013. Springer Berlin Heidelberg. [Cited on pages 26, 87, 104, 107, and 109]

[ZMH+18] Jan Henrik Ziegeldorf, Roman Matzutt, Martin Henze, Fred Grossmann, and Klaus Wehrle. Secure and anonymous decentralized bitcoin mixing. *Future Generation Computer Systems*, 80:448 – 466, 2018. [Cited on page 141]

Appendix A
Privacy in Cryptocurrencies

In Chapter 3 and Chapter 4, we explain kleptographic and steganographic attacks in blockchains. Both types of attacks are implementation attacks that exploit the uncontrolled randomness in cryptographic primitives in blockchains. These primitives are mainly used to achieve privacy in blockchains. Therefore, this appendix presents an overview of the different tiers of privacy guarantees in cryptocurrencies, and the used anonymity techniques.

Existing surveys on blockchains consider general security issues and challenges in Bitcoin and cryptocurrencies without any particular focus on anonymity and privacy [CSLR18, LJC+17, DSG19, JHW18]. The only existing survey that focuses on privacy is the work of Khalilov et al. [KL18]; however, it does not provide any classification of the different levels of anonymity in cryptocurrencies, nor does it explain the related anonymity techniques. On the contrary, this appendix, which mainly reflects our published survey [AZ19a], provides a systematic study on anonymity in blockchains. The content of this appendix can be summarised as follows. Sec. A.1 presents a novel categorization for the tiers of anonymity offered in the diverse cryptocurrencies. Sec. A.2 examines the techniques used to achieve the different tiers of anonymity and highlights their known vulnerabilities and weaknesses. Finally, Sec A.3 compares the anonymity techniques and attempts to forecast their technological trends.

A.1 Tiers of Anonymity in Cryptocurrencies

The offered anonymity in any cryptocurrency and blockchain application can be assessed by considering two characteristics: (1) the ability of the used anonymity scheme to break any possible linkage between transactions, and (2) its ability in hiding users' identities (senders and receivers). Given these two characteristics, we define the following three tiers of anonymity in cryptocurrencies: (1) *pseudonymity*, (2) *set anonymity*, and (3) *full anonymity*. Below we describe each of these tiers:

1. **Pseudonymity** is the most primitive level of anonymity in cryptocurrencies. In pseudonymity, the users' identities are hidden using pseudo-anonymous addresses. For example, this is the level of anonymity guaranteed in Bitcoin.
2. **Set anonymity** is when the identity of the user is either 1 out of n possible identities. Set anonymity is achieved by using ring signatures [RST01] where n is equal to the size of the ring. For instance, the CryptoNote framework and its cryptocurrencies

use linkable ring signatures to achieve set anonymity. Similarly, mixers provide set anonymity where n is equal to the number of inputs in the mix.

3. **Full anonymity** is provided when the sender can be any node/entity in the blockchain, and the sent note or coin can be any unspent coin. As shall be discussed, this level is attained by using commitments and zero-knowledge proofs as in Zerocoin [MGGR13] and Zcash [HBHW18].

Besides, there is a very notable feature that is used by some cryptocurrencies in conjunction with these three levels of anonymity. We refer to this feature as ***confidential transactions*** that ensures the transacted amounts are hidden. We consider *confidential transactions* as an anonymity feature rather than a separate level of privacy since a cryptocurrency can, for example, guarantee *set anonymity* while offering confidential transactions at the same time, as is the case in Monero [mon19]. We emphasize that we do not limit this feature to Monero's confidential transactions [Max], and include any approach that hides or obfuscates the transferred amounts to thwarts transaction flow analysis.

Fig. A.1 shows a pie chart representing the distribution of 20 currencies and implementations according to their tier of anonymity. Details of these currencies are shown in Table 1.1 in Capter 1.

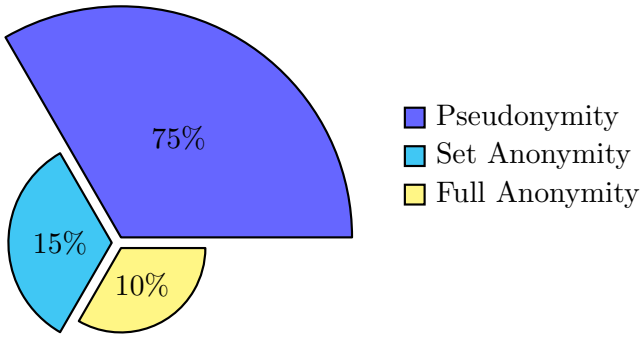


Figure A.1: Distribution of 20 cryptocurrencies and protocols according to their tier of anonymity.

A.2 Privacy Techniques in Cryptocurrencies

To achieve the three anonymity levels and confidential transactions as aforementioned in Sec. A.1, cryptocurrencies implement numerous anonymity techniques. In this section, we briefly discuss six major techniques: **(1)** pseudonymous addressing, **(2)** ring signatures, **(3)** mixers, **(4)** commitments, **(5)** zero-knowledge proofs, and **(6)** stealth addressing. Besides, we give examples of the cryptocurrencies that implement each of these techniques and list the known attacks and weaknesses concerning each technique.

A.2.1 Pseudonymous addressing

Pseudonymous addressing aims to preserve privacy by breaking the link between addresses and their owners' real identities. It is widely known that Bitcoin is an example implementation of *pseudonymous addressing* in cryptocurrencies.

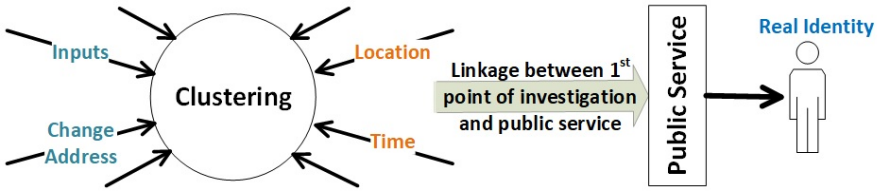


Figure A.2: Representation of clustering attacks on Bitcoin.

Pseudonymity in Bitcoin. Bitcoin is the first cryptocurrency and remains to be the most successful one with a market value of over $211 billion[1], and more than 9000 participating nodes[2]. As explained in its whitepaper [Nak08], Bitcoin preservers users' anonymity through the use of pseudonymous addresses; a user's address is the Base-58 encoding of the following 25-byte binary string:

$$\begin{aligned} \epsilon &= \mathsf{RIPEMD\text{-}160}(\mathsf{SHA\text{-}256}(\mathsf{publicKey})) \\ \mathsf{address} &= \mathsf{v}||\epsilon||\mathsf{checksum} \end{aligned}$$

Where publicKey is the user's public key, v is a one-byte value indicating the version, and checksum is the least significant four bytes of the following value: $\mathsf{checksum} = \mathsf{SHA\text{-}256}(\mathsf{SHA\text{-}256}(\epsilon))$ [TS16, Wik18]. Furthermore, although they do not entirely enhance anonymity [AKR+13], Bitcoin users are advised to take two protective measures. Firstly, users can generate a new key pair for each transaction [Nak08]. In practice, a person generating a transaction will also generate a new key pair so that the *change address* is not linked to the originating address and is indistinguishable from the destination's address. Secondly, in every transaction, the sender fully empties one or more accounts, *inputs*, and creates one or more accounts, *outputs*. This approach helps break the linkage between the user's accounts [But16]. Nonetheless, addresses can still be clustered, and the real identity of the users can eventually be revealed using the aid of public services, as demonstrated in many publications [MPJ+13, RH11, RS13, TS16].

Attacks on pseudonymous addressing. *Pseudonymous addressing* provides a weak anonymity guarantee. In fact, it is mentioned on Bitcoin's official website that Bitcoin is *not* anonymous [bit19b]. Consequently, it is not surprising that various de-anonymization attacks have been proposed in the relevant literature. Overall, these attacks can be classified into two broad categories: (1) clustering and Bitcoin blockchain analysis, and (2) the exploitation of the Bitcoin P2P network and diffusion protocol.

As depicted in Fig. A.2, clustering attacks analyse Bitcoin traffic flow to cluster a given user's addresses and transactions and link them to a public service, e.g. an exchange website. These services can then reveal the real identity behind the pseudonymous addresses. In general, there are two approaches to clustering: analysis of transactions' inputs and outputs [MPJ+13, RH11, RS13, TS16], and behavioural analysis of attributes like time, location, and spending habits [AKR+13, RS14, DS15].

The second category of attacks on pseudonymous addressing is through the exploitation of the Bitcoin P2P network. This set of attacks was pioneered by the work of Koshy et al. [KKM14], who studied the relay patterns of transactions and were able to map

[1] `https://coinmarketcap.com/` on 08/08/2019
[2] `https://bitnodes.earn.com/` at 13:22:21 UTC on 08/08/2019.

between 252 and 1162 Bitcoin addresses to the IP addresses that likely own them. For more details on these attacks, a summary is included in Table A.1.

Attacks on Pseudonymous Addressing
1) Clustering and Bitcoin blockchain analysis.
As shown in Fig A.2, this attack analyses Bitcoin traffic flow to cluster the user's addresses and transactions and link them to a public service, e.g. an exchange website. These services can then reveal the real identity behind the pseudonymous address. In general, there are two approaches to clustering: **(i)** analysis of transactions' inputs and outputs, and **(ii)** behavioural analysis.
Analysing transactions' inputs and outputs. Use two particular heuristics to cluster transactions: **(i)** The first heuristic is that *inputs in one transaction are likely to be owned by the same user* [Nak08]. Different users can theoretically contribute inputs in the same transaction but *rarely* do so. **(ii)** The second heuristic is that *the newer output address can be assumed to be the change address of the user who generated the transaction* [MPJ+13, RH11, RS13, TS16]. These heuristics result in representations of Bitcoin addresses, transactions, and users/entities. The last step in traffic analysis is to map users from these representations to real-world identities. This step aims to establish *ownership* and can be accomplished using the aid of public services, e.g. online stores and exchange website, which usually have user-identifying information such as email addresses and even bank accounts [RH11]. To demonstrate the effectiveness of the above heuristics, the authors of [MPJ+13], attempted to track known Bitcoin thefts to exchange services. In summary, they were able to track 6 out of 7 thefts from the point of theft to an exchange service. Furthermore, if presented with legal subpoena, exchange services can reveal the real identities behind the exchange operation. More importantly, their success in tracking these thefts prove that even privacy-conscious users, who seek to further hide their identities by sending (or peeling) some of their funds to newly generated addresses, are prone to de-anonymization using these heuristics.
Behavioural Analysis. In this type of analysis, addresses and transactions are clustered based on behavioural attributes like their time, location, and amount. Androulaki et al. [AKR+13] used behavioural analysis to augment their clusters. Namely, they considered the time of the transactions, the indices of the inputs in a transaction, and the transferred amount. Using these attributes, they succeeded in unveiling about 40% of the users in their simulated Bitcoin network. Similarly, Ron et al. [RS14] used behavioural analysis to link the Bitcoin addresses that are believed to be related to the Silk Road marketplace [mar19]. Also, Dupont et al. [DS15] analysed the user's spending habits to reveal the physical location of Bitcoin users. They assessed their method by collecting 518 known charities' Bitcoin addresses and physical locations, and comparing this data against their informed guesses, where their initial results show an accuracy of up to 72%.
2) Exploiting Bitcoin P2P Network.
This family of attacks exploits the nature of the Bitcoin P2P network to link pseudonymous addresses to IP addresses. The work of Koshy et al. [KKM14] constitutes the first proposal to de-anonymize Bitcoin users by studying the relay patterns of transactions, and they were able to map between 252 and 1162 Bitcoin addresses to the IP addresses that likely own them[3]. Similarly, the work in [JSKV18] attempts to develop a probabilistic model to identify transactions' originators' IP based on monitoring the nodes that first relay a given transaction.

[3] Linking Bitcoin pseudonyms to the user's IP does not only cause a privacy breach but may also allow attackers to launch DoS against that user's IP. This is more relevant if this user is a vendor or a service provider.

Table A.1 Continued from the previous page
Moreover, the authors of [BKP14] attempted to de-anonymize Bitcoin clients, even those sitting behind NATs, by the set of *entry nodes* they connect to. According to their methodology, the attacker tries to connect to the majority of servers, and they argue that when the attacker receives the transaction from 2 to 3 entry nodes, he can map the transaction to a specific client with a very high probability.[4] To strengthen their anonymity, Bitcoin users may choose to use anonymization tools, such as Tor [tor19]; however, as shown by Biryukov et al. [BP15], combining Bitcoin and Tor introduces a new attack vector. Biryukov et al. [BP15] explored the exploitation of Bitcoin P2P with Tor beyond the mere banning of Bitcoin clients from using Tor *exit nodes* as previously done in [BKP14]. Their attack depends on: (1) forcing Bitcoin clients to connect to the attacker's Tor exit nodes or directly to the attacker's Bitcoin peers, and (2) fingerprinting clients by writing unique, possibly fake, addresses to the target's address table. In 2015, the Bitcoin community responded to the aforementioned attacks by changing its transactions broadcasting protocol from a gossip-like *trickle spreading* protocol to a *diffusion spreading* protocol [Wui15]. To assess the impact on anonymity, the authors of [FV17a] studied the properties of the two broadcasting protocols and their effect on user anonymity. They concluded that the two Bitcoin flooding protocols do not protect user anonymity. Also, Mastan et al. attempted to de-anonymize Bitcoin users sitting behind Tor by studying the pattern of their sessions and constructing a *session graph*, which they were able to perform with a precision of 0.9 [MP18] .
3) Other attacks
Goldfeder et al. [GKRN18] studied the effect of web trackers on Bitcoin users when shopping online. They concluded that trackers could uniquely identify transactions, link them to the user's cookie, and reveal the user's real identity. Other tools and frameworks have been proposed for visual traffic analysis of the Bitcoin blockchain [SMZ14, BDP$^+$15].

Table A.1: Attacks on Bitcoin pseudonymity

A.2.2 Ring signatures

Ring signatures are explained in detail in Sec. 2.2.6.2, and we discuss in the following the known attacks on ring signatures.

Attacks on ring signatures. Although ring signatures have evolved over time, there remain some weaknesses that have been exploited to reveal the sender's identity/index. One of the first attacks on ring signature was the deducibility of the real consumed coin as a result of referencing outputs that have been provably spent or consumed in 0-mixin transactions [MSH$^+$18]. More known attacks on ring signatures are listed in Table A.2.

[4] A list of current active Tor exit nodes can be found in: `https://torstatus.blutmagie.de/`

Vulnerabilities and Weaknesses of Ring Signatures	
Weakness	**Description**
Deducibility due to 0-mixin coins.	The authors of [MSH+18] described two weaknesses in Monero's ring signature. The first weakness is the deducibility of real spent input as a result of referencing outputs that have been provably spent or consumed in previous 0-mixin transactions. In other words, 0-mixin transactions do not only de-anonymize the output they reference but also de-anonymize other transactions with mixin ≥ 1.
Identifying real inputs using temporal analysis.	The second weakness described in [MSH+18] is related to the sampling of mixins, or decoy coins. Namely, they have found that about 80% of the time, the real consumed output is the newest created coin.
Flooding the network with the attacker-generated outputs.	The authors of [WLSL18] described two attacks on the Monero ring signature. The first attack is an extension to the discussion from [NNM14] and based on flooding the network with outputs that are generated by the attacker(s) and addressed to their own addresses. Therefore, if these outputs are referenced as decoy outputs, i.e. mixins, in any ring signature, the attacker(s), who passively monitors the signatures, can rule out their outputs and hence decrease the anonymity of the signer. If, for example, a transaction references k outputs/coins, i.e. has a mixin of size k, and t of which are generated by the attacker, then the effective mixin size is reduced to $(k - t)$.
Subverted sampling of outputs.	The second attack described in [WLSL18] is an active version of the previous attack. Namely, the attacker mis-implements wallets to sample his outputs when generating ring signatures. Therefore, the attacker who continuously monitors all transactions could de-anonymize the real spent outputs.
Anonymity reduction by observing identical UPID.	The authors of [WLS+18] described an anonymity reduction attack on Monero transactions by observing identical UPID in different transactions. Namely, they state that if a transaction T_a has a UPID U_a and generates some output O_a, and a latter transaction T_b that uses the same UPID U_a and references O_a as part of its mixins, then O_a is likely to be the real spent output in T_b and not a mere decoy output.
De-anonymization by new forks	Wijaya et al. [WLS+19] demonstrated that Monero hard forks could lead to traceability of the real spent outputs when the user spends the coins in the original blockchain and the newly forked blockchain.
Time and size	Borromean ring signatures, which were used to construct *rangeproofs* in Monero RingCT resulted in rangeproofs that are several kilobytes in size and take milliseconds to verify [Poe18]. Hence, the crypto community has been looking for a more succinct and faster to verify, which eventually resulted in devising *Bulletproofs* [BBB+18] as discussed in Sec. A.2.5.

Table A.2: List of attacks on ring signatures

A.2.3 Mixers

To address the privacy limitations in Bitcoin, there have been multiple proposals to break any linkage between senders and recipients by mixing users' funds through coin-laundry services called *mixers*. These mixers are generally in the following three forms. **(1)** *Trusted centralized mixers* were the 1^{st}-generation of mixers and demand *unreasonable* trust of third-party services to mix the user's coins. This approach has a single point of failure,

Type	Examples	Disadvantage
Centralized Mixers	CryptoMixer [cry19] Bitcoin Fog [bit19a] BestMixer [bes19] Mixcoin [BNM+14] Blindcoin [VR15]	Single point of failure
		No deniability against the mix itself [ZGH+15]
		No proof of mixing
		Unreasonable trust of 3^{rd} party
		Possible theft
	Obscuro [TLK+18]	Uses trusted execution environments (TEE) Assumes no mis-implementation by mixer operator Anonymity set is limited by block size [TLK+18]
Smart-contract-like Mixers	CoinJoin⋆ [Max13a] CoinShuffle† [RMSK14] Coinswap [Max13b]	⋆No anonymity against insiders, i.e. users in the mix
		Vulnerable to Sybil attacks
		Vulnerable to collusion between users in the mix
		Anonymity set = the number of users in the mix [ZGH+15]
		†last user determines the outcome of the shuffle [ZGH+15]
		Malicious users can disrupt mixing [TLK+18][1]
Decentralized Mixers	CoinParty [ZGH+15]	Longer mixing delay [ZGH+15]
		Assumption 2/3 of the peers are honest
	Zerocoin [MGGR13]	Anonymity level is related to number of minted coins (between a coin's mint and its spend) [MGGR13]
		Reveals the number of minted and spent coins [MGGR13]
		Reveals transferred denominations [MGGR13]

Table A.3: List of proposed mixers in literature. 1: Users can join mixing and then abort to disrupt the operation

i.e. the trusted mixer, and does not provide any proof of mixing. **(2)** *Smart-contract-like mixers* in which multiple users agree to create a joint transaction to obfuscate inputs and outputs, e.g. CoinJoin [Max13a] and CoinShuffle [RMSK14]. In general, these mixers do not provide anonymity against other users in the mix and are vulnerable to collusion between users in the mix. **(3)** *Decentralized mixers* are trust-free cryptographic extensions to Bitcoin, e.g. Zerocoin [MGGR13] and CoinParty [ZGH+15, ZMH+18]. It is important to note that this type of mixers does not represent fully-fledged anonymous zero-knowledge-proof currencies, which are discussed later in Sec. A.2.5. Instead, this type represents extensions on top of other currencies, and may not be practical for day-to-day usage. For example, Zerocoin [MGGR13] is presented as a decentralized mix that extends Bitcoin; however, its limited functionality and high computational cost do not allow it to be used for routine transactions. For more details, Table A.3 provides an up-to-date list of mixers proposed in the literature and their known vulnerabilities.

A.2.4 Commitments

Commitments are widely used as references or pointers to some secret, which allow the owner of the secret information to demonstrate its properties without revealing the secret information. For instance, the user could commit the balance of his bank account, and then use zero-knowledge proofs to show the balance is within a certain range, e.g. larger than 0. A commitment scheme $\mathcal{CS}$ typically consists of three algorithms: $\mathcal{CS} = (\mathsf{Setup}, \mathsf{Commit}, \mathsf{Open})$, as follows [ADR02]:

- $\mathsf{ck} \leftarrow \mathsf{Setup}(1^\lambda)$. On the input of a security parameter 1^λ, the setup algorithm generates a commitment key ck.
- $c \leftarrow \mathsf{Commit}_{\mathsf{ck}}(m, r)$. The commitment algorithm takes as input a message m from the message space $\mathcal{M}$, to which the user is committing, and a random coin r. The

commitment algorithm Commit outputs a commitment c.

- $m \leftarrow \mathsf{Open}_{\mathsf{ck}}(c, r)$. The open algorithm reveals the committed message m if provided with a valid (c, r) pair, and outputs $\perp$ otherwise.

It is required that the statement $m \leftarrow \mathsf{Open}_{\mathsf{ck}}(\mathsf{Commit}_{\mathsf{ck}}(m, r), r)$ is true except for a negligible probability ϵ. Besides, all commitment schemes should satisfy two security properties: *hiding* and *binding*. Hiding means that the commitment c does not reveal any information about the message it is committed to, m. In other words, there is no PPT adversary $\mathcal{A}$, who distinguishes with a non-negligible probability if a commitment c is a commitment to m_0 or m_1, where the two messages are provided by $\mathcal{A}$. Whereas, binding means that it is computationally infeasible for an adversary $\mathcal{A}$ to find two distinct messages m_0 and m_1 that commit to the same value c [ADR02].

In the context of blockchains, two types of commitment schemes have been used: **(i)** additive homomorphic commitments such as Pedersen commitment, and **(ii)** non-malleable commitments like hash-based commitments.

(Generalized) Pedersen commitment [Ped92] is used in many blockchain platforms, such as Monero. To make the commitment non-interactive, the commitment key is typically given as a common reference string. Let $(G, H) \in \mathbb{G}^2$ be the commitment key. To commit a message $m \in \mathbb{Z}_q$, the committer picks a fresh randomness $r \in \mathbb{Z}_q$ and outputs the commitment as $c = \mathsf{Commit}(m, r) = rG + mH$. This commitment is said to be computationally *binding* and unconditionally *hiding*. In Monero, the group is instantiated from the elliptic curve (the $\mathsf{secp256k1}$ curve). In addition, *Pedersen commitments* are additively homomorphic, i.e. they preserve addition and commutativity, which enables the public verification that the sum of the hidden input value(s) is equal to the sum of the hidden output value(s). For example, disregarding the transaction fee for simplicity, if a transaction has three inputs a, b, and d, and two outputs e and g such that $a + b + d = e + g$, then $\mathsf{Commit}(a + b + d, r) = \mathsf{Commit}(e + g, r) = \mathsf{Commit}(a, r_1) + \mathsf{Commit}(b, r_2) + \mathsf{Commit}(d, r_3) = \mathsf{Commit}(e, r_4) + \mathsf{Commit}(g, r_5)$, given $r = r_1 + r_2 + r_3$ and $r = r_4 + r_5$.

Hash-based commitments are used in Zcash due to their efficiency, which enables fast zk-SNARK. Unlike the Pedersen commitment, hash-based commitments are usually transparent in the sense that the setup process is a public coin. Hence, unlike common reference-string-based schemes, the setup process is believed to be subversion resistant. To commit a message $m \in \{0,1\}^*$, the committer picks a random coin $r \in \{0,1\}^\lambda$ and outputs the commitment as $c = \mathsf{hash}(m, r)$, where λ is the security parameter, e.g, 256. In Zcash, the hash function was instantiated from $\mathsf{SHA\text{-}256}$, and recently switched to a group-based structure-preserving hash. More details can be found in the Zcash protocol specification [HBHW18].

A.2.5 Non-interactive zero-knowledge proofs

Typically, the zero-knowledge proofs used in blockchains need to be publicly verifiable, which means they need to be non-interactive. On the other hand, it is well-known that non-interactive zero-knowledge (NIZK) proofs cannot be realized in the standard model, also known as the plain model. Therefore, all of the non-interactive zero-knowledge proofs require some setup assumptions, such as common-reference string, random oracle, etc. In general, NIZK proofs have been used to achieve anonymity in cryptocurrencies in three ways:

1. Utilizing the existing scripts in current cryptocurrencies to extend these cryptos and break the linkage between the senders and the receivers. Zerocoin [MGGR13] is an example of this methodology.
2. Devising new cryptographic structures to replace current inefficient structures. An example of this type is the use of Bulletproofs [BCC+16] to replace the Borromean ring signatures in Monero RingCT's rangeproofs.
3. Designing new ZKP-based cryptocurrencies that are fully anonymous like Zcash [zca20], which is an implementation of the Zerocash protocol [SCG+14].

In the following, we discuss two commonly used NIZK schemes.

zk-SNARK. Succinct non-interactive zero-knowledge argument of knowledge (zk-SNARK) has two very fundamental properties: (i) succinctness, and (ii) being unbalanced. *Succinctness* means that the proof size is less than poly-logarithmic, or constant in this concrete case, with respect to the witness size. Being *unbalanced* indicates that the verifier's running time is much less than the statement execution time, i.e. poly-logarithmic. In blockchains, a zero-knowledge proof needs to be verified by a great number of verifiers; hence, unbalanced proofs are preferred. However, the cost of the proof generation is usually very high, which limits its wide adoption.

zk-SNARK is used to achieve full anonymity in Zerocash [SCG+14], which is a digital currency that is decentralized, privacy-preserving, and efficient. To anonymize the sender and the receiver, and to mask the transferred amount, Zerocash uses zk-SNARK. Zcash [zca20] is a cryptocurrency that implements the Zerocash protocol. The Zcash blockchain contains two sets: a set of all commitments cm, and a set of all created nullifiers nf. Hence, the Zcash blockchain does not only contain a database of unspent *transactions* but a database of all *transactions* that ever existed. To each *note*, there is a cryptographically associated *note commitment* and a *nullifier*, i.e. there is a 1:1:1 relation between *notes*, *note commitments*, and *nullifiers*. Computing the *nullifier* requires the associated private *spending key* $\mathsf{a_{sk}}$. It is infeasible to correlate the *note commitment* with the corresponding *nullifier* without knowledge of this spending key. An unspent valid *note*, at a given point on the blockchain, is one for which the *note commitment* has been publicly revealed on the blockchain prior to that point, but the *nullifier* has not.

The basis of the privacy properties of Zcash is that when a *note* is spent, the spender only proves that some commitment for it have been revealed, without revealing which one. This implies that a spent *note* cannot be linked to the transaction in which it was created. That is, from an adversary's point of view, the set of possibilities for a given note input to a transaction includes all previous notes that the adversary does not control or know to have been spent.

Bulletproofs. Bulletproofs are shorter zero-knowledge proofs that were proposed by Bünz et al. [BBB+18] and are based on the work of Bootle et al. [BCC+16]. While zk-SNARK requires the use of bilinear groups and pairing-based cryptography, Bulletproofs are based on discrete-log computation. Hence, Bulletproofs are suitable for all elliptic-curve algorithms and can prove arbitrary arithmetic circuit. For the prover's running time, Bulletproof is much faster than zk-SNARK; however, the verifier's running time is typically similar to the prover's running time, which is linear in the statement execution. It means that Bulletproofs cannot be used to achieve verifiable computation sourcing, as the verifier needs to spend an equal amount of time to verify the proof. In Monero, Bulletproofs substantially reduce the size of transactions by replacing the Borromean ring signature [MP15] in generating *rangeproofs*.

Disadvantages of zero-knowledge proofs. Zero-knowledge proofs (ZKP) can provide strong anonymity guarantees, as in Zcash and Zerocoin; however, they suffer from few disadvantages. First, it is proven that non-interactive ZKPs for general NP language must require some trusted setup assumptions, such as the security parameter's generation ceremony in Zcash. Also, the proof generation and verification can be computationally inefficient. More importantly, the prover's efficiency is far from being practical for large-scale statement or verifiable computation.

A.2.6 Stealth addressing

Stealth addressing is a technique proposed as part of the CryptoNote protocol [Sab13] to hide the recipient's identity (or address). In short, the sender, Alice, uses Diffie-Hellman exchange [DH76] to compute a shared secret and generate a one-time destination address that can only be identified by the intended recipient, Bob. Specifically, let us assume that Bob's public key is the pair (A, B) that corresponds to his private key (a, b), such that $A = aG$, and $B = bG$, where G is the base point of the used elliptic curve. In this case, Alice generates a random number r, and a one-time address $P = \mathcal{H}_s\{rA\}G + B$, where $\mathcal{H}_s$ is a collision-resistant cryptographic hash function. Along with P, Alice sends $R = rG$ as part of the transaction. Bob checks every transaction using his private key (a, b) and computing $P^{'} = \mathcal{H}_s\{aR\}G + B$. If the transaction is destined for Bob, then $P^{'} = P$. More generally, the stealth addressing technique can be generalised to any non-interactive key exchange (NIKE) together with the public key system. Hence, the above scheme can be easily extended to a post-quantum secure stealth addressing scheme by replacing the underlying primitives.

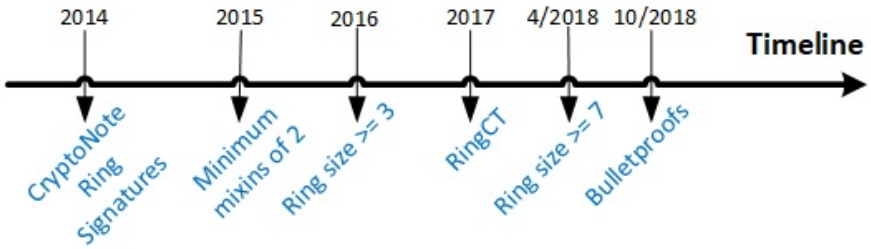


Figure A.3: Timeline of Monero privacy enhancements [A. 15, mon19].

A.3 Summary of Privacy in Cryptocurrencies

Summary. Table 1.1 summarizes the tiers of anonymity offered in 20 currencies and the techniques they implement to achieve privacy and anonymity. As shown in Table 1.1, one can conclude that ZKP and commitments are used to achieve the highest tier of anonymity, i.e. full anonymity. However, these two techniques can increase the computational cost and may require a trusted setup. Therefore, anonymity schemes should not be assessed only by considering the level of anonymity they provide but by jointly examining three factors: (1) the level of *anonymity* provided by using the scheme, (2) the scheme's computational *efficiency*, and (3) the extent of the needed *trust* to use the technique.

Technology trend. The future technological trend in privacy mechanisms is best exemplified by Monero's evolution since its inception. As shown in Fig. A.3, Monero was initially based on the CryptoNote protocol, which uses linkable ring signatures, and then evolved over time and adapted the use of Bulletproofs to replace the Borromean ring signature in its RingCT's rangeproofs. This demonstrates the continuous quest for cryptocurrencies to adapt privacy schemes that: (1) offer a higher tier of anonymity, (2) require less generation and verification time, (3) produce more succinct proofs, (4) do not require trusted setup, and (5) possibly, result in lower transaction fees[5]. The design of new anonymity schemes should also consider their impact on the forkability of the cryptocurrency [WLS+19].

Open problem. There are some known intrinsic vulnerabilities concerning anonymity in cryptocurrencies and blockchains in general. One of these vulnerabilities is leaking the user's IP address and timestamp whenever a user broadcasts a transaction. This can be exploited, as demonstrated in many works [KKM14, BKP14, BP15, FV17a], to de-anonymize the users regardless of the specific blockchain application they are using. Furthermore, even when using an anonymization tool, like Tor, users can still be de-anonymized as detailed in Table A.2.

[5] Usually, transaction fees are inversely proportional to the size of the proofs. For example, when Monero adopted the use of Bulletproofs, their average transaction fees decreased by more than 90% [O'L19].

Appendix B
Case study: Bytecoin Wallet Bug

In this section, we explain an implementation bug that we discovered while implementing our kleptographic attack in Bytecoin's wallet. Although this bug is not itself a *kleptographic* attack, it highlights how the actual implementation of cryptographic primitives could diverge from their specifications without being noticed by end-users. This bug demonstrates the plausibility of unseen kleptographic attacks.

As previously discussed in Sec. 2.2.6.2, CryptoNote-based cryptocurrencies use a linkable ring signature to enhance the senders' privacy and achieve set anonymity, as explained in Appendix A. As such, the signer's public key is hidden among a set of randomly sampled public keys. To obscure the identity of the actual signer, the signer's index within the ring is kept secret, and the signer's public key should be indistinguishable from other keys in the ring.

While experimenting with Bytecoin's wallet, we observed a significant discrepancy between the wallet execution behaviour and the above-mentioned CryptoNote specifications. Peculiarly, the CryptoNote specifications require the signer's index within the ring to be randomly picked so that the signer's public key is randomly placed in the ring. On the contrary, Bytecoin tends to put the signer's public key as the last key in the ring. This bug effectively nullifies the use of the ring signature.

Table B.1 shows the statistical details of our experiments with the Bytecoin wallet. The size of the ring k is set to a value between 3 and 10, and in each case, the function responsible for generating the ring signature, generate_ring_signature(...), is invoked 10000 times. The table shows the percentages when the secret index ℓ is the last in the ring, i.e. when $\ell = k - 1$, also when $\ell = k - 2$ and $\ell = k - 3$. It can be seen that the probability of $(\ell = k - 1)$ increases as the size of the ring decreases. In addition, it can be observed that the probability is about 98% that the secret index ℓ is greater than or equal to $(k - 3)$.

This bug effectively diminishes the set anonymity promised by CryptoNote's ring signature and facilitates blockchain analysis attacks. Hence, we reported the bug to Bytecoin's developers, who replied to our report on 10 July 2018 and acknowledged the issue and their ongoing effort to rectify it. Note that this bug still exists in the latest version of Bytecoin (v 3.5.1) although the distribution of the secret index is biased towards the lower values, e.g. given k is 10, we found that $\ell \in \{0, 1, 2, 3\}$ in $\geq 90\%$ of the time.

Experiments on Bytecoin Wallet				
k	$\ell = k-1$ (%)	$\ell = k-2$ (%)	$\ell = k-3$ (%)	$\ell \geqq k-3$ (%)
3	86.61	12.85	0.54	100
4	81.11	17.39	1.45	99.95
5	76.82	20.61	2.45	99.88
6	70.23	25.28	4.15	99.66
7	66.43	27.64	5.31	99.38
8	61.2	30.48	7.17	98.85
9	57.09	33.05	8.24	98.38
10	54.37	33.39	10.07	97.83

Table B.1: Experimenting with Bytecoin wallet (v 3.0.0) with different ring sizes k, and the percentages show when the secret index ℓ is either one of the last indices in the ring. According to specifications, ℓ should be picked randomly, i.e. $\ell \xleftarrow{\$} [0, k-1]$.

Appendix C
Security Proofs

This appendix contains the formal security proofs for some of the kleptographic attacks, steganographic tools, and cryptographic schemes in this thesis. Sec. C.1 describes the proof of the security, i.e. computational undetectability, of our kleptographic attack on CryptoNote's ring signature. Sec. C.2 presents our proof for the undetectability of Skywhisper based on the pseudo-randomness of its Encrypt function. Moreover, in Sec. C.3, we prove the existential unforgeability of our re-randomized ring signature RRS, which is described in Sec. 6.1.1. In Sec. C.4, we show the dis-aggregation property of Boneh et al.'s aggregate signature [BGLS03] that is mentioned in Sec. 6.2. Finally, Sec. C.5 proves the existential unforgeability of our re-randomizable and aggregatable signature RAS proposed in Sec. 6.3.

C.1 Security proof of Theorem 3.1

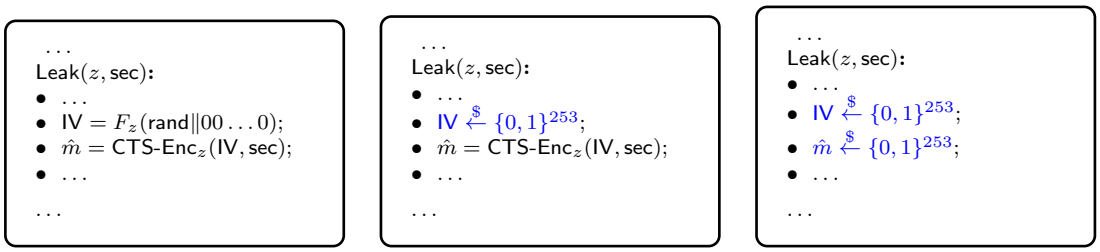
The security of the proposed kleptographic attack on CryptoNote is examined for undetectability under the security game in Fig. 3.3. Informally, if there is not any PPT watchdog $\mathcal{W}$ that can distinguish between the output of a subverted ring signature and that of the original signature except for a negligible probability, then the proposed kleptographic attack is undetectable and is said to be secure.

Theorem 3.1 states that if both F and CTS-Enc*, as shown in Fig. 3.6 are a secure pseudo-random function (PRF) and a semantically-secure encryption algorithm, respectively, then the kleptographic attack described in Fig. 3.6 is undetectable with respect to any PPT $\mathcal{W}$.*

The advantage of a PPT watchdog $\mathcal{W}$ in detecting subverted signatures, and thus his probability in winning the kleptography security game of Fig. 3.3 is given as follows:

$$\mathsf{Adv}^{\mathsf{SU}}_{\mathcal{W}}(1^\lambda) = \left| \Pr\left[\mathbf{Expt}^{\mathsf{SU}}_{\mathcal{W}}(1^\lambda)\right] - \frac{1}{2} \right| = \mathsf{negl}(\lambda)$$

Proof. We prove Theorem 3.1 using hybrid proofs and reduction. In particular, Fig. C.1a represents a kleptographically-subverted CryptoNote's ring signature, as detailed in Fig. 3.6, whereas Fig. C.1c is equivalent to the original non-modified CryptoNote's ring signature. Besides, Fig. C.1b is an intermediate step between Fig. C.1a and Fig. C.1c. We aim to prove the computational indistinguishability between the kleptographically-subverted algorithm in Fig. C.1a and the original algorithm in Fig. C.1c. In order to do



(a) H1 **(b)** H2 **(c)** H3

Figure C.1: H1 represents the kleptographically-modified CryptoNote's ring signature, which is shown in detail in Fig. 3.6. H2 represents an intermediate hybrid, and H3 represents the original non-modified CryptoNote's ring signature. Our goal is to prove Theorem 3.1 by demonstrating that H1 and H3 are computationally-indistinguishable if F is a secure PRF function, and CTS-Enc$_z$(IV, sec) is a semantically-secure encryption algorithm.

so, we will show that the subverted algorithm in Fig. C.1a is indistinguishable from the intermediate hybrid algorithm in Fig. C.1b if F is a secure pseudo-random function (PRF). After that, we will show that Fig. C.1b is indistinguishable from the original algorithm in Fig. C.1c if CTS-Enc is a semantically-secure encryption algorithm. Consequently, we conclude that Fig. C.1a and Fig. C.1c are indistinguishable if F and CTS-Enc, as shown in Fig. 3.6, are a secure pseudo-random function (PRF) and a semantically-secure encryption algorithm, respectively, which proves Theorem 3.1.

First: proving that H1 and H2, in Fig. C.1a and Fig. C.1b, respectively, are indistinguishable if F is a secure PRF. We use reduction to prove the indistinguishability between H1 and H2. In particular, we assume that there exists a PPT $\mathcal{W}$ who can distinguish between the output of H1 and H2 with a non-negligible probability. Then, we construct a PPT adversary $\mathcal{A}$ who can break the PRF game of F, i.e. $\mathcal{A}$ must be able to distinguish with a non-negligible probability between the pseudo-random text generated by F and randomly-selected text, as explained in Sec. 2.2.3.

Let us denote by $\mathcal{C}$ the challenger, who randomly picks $b \xleftarrow{\$} \{0,1\}$. If $b = 0$, $\mathcal{C}$ executes H1, i.e. IV $= F_z(\mathsf{rand}\|00\ldots0)$, and if $b = 1$, $\mathcal{C}$ executes H2, i.e. IV $\xleftarrow{\$} \{0,1\}^{253}$. In both cases, the generated signature σ, which contains IV in its random numbers (c, r)'s as detailed in Fig. 3.6, is passed to $\mathcal{A}$ who is challenged to break the PRF of F by distinguishing if IV is sampled randomly or generated using F. After receiving σ, $\mathcal{A}$ passes it to $\mathcal{W}$ who is challenged to distinguish whether or not σ was generated using H1 or H2. $\mathcal{W}$ outputs its guess as $b^* = 0$ to indicate σ is generated by H1 and $b^* = 1$ otherwise. After receiving $\mathcal{W}$'s output b^*, $\mathcal{A}$ outputs its guess b' based on b^*, particularly, $b' = b^*$. Since $\mathcal{W}$ can distinguish between the output of H1 and H2 with a non-negligible probability, and $b' = b^*$, then $\mathcal{A}$'s advantage in breaking the PRF of F, denoted by $\mathsf{Adv}^{\mathsf{PRF}}_{\mathcal{A}}$, is also non-negligible. *Hence, we conclude that H1 and H2 are computationally indistinguishable if F is a secure PRF function.*

Second: proving that H2 and H3, in Fig. C.1b and Fig. C.1c, respectively, are indistinguishable if CTS-Enc is a semantically-secure encryption algorithm. Similar to the first part of the prove, we use reduction to prove the indistinguishability between H2 and H3. We assume there exists a PPT $\mathcal{W}$, who can distinguish between H2 and H3 with a non-negligible probability, and construct a PPT adversary $\mathcal{B}$, who can break the semantic security of CTS-Enc, i.e. $\mathcal{B}$ must be able to distinguish with

a non-negligible probability between the pseudo-random ciphertext generated by CTS-Enc and randomly-selected text.

Let us denote by $\mathcal{C}$ the challenger, who randomly picks $b \xleftarrow{\$} \{0,1\}$. If $b = 0$, $\mathcal{C}$ executes H2, i.e. $\hat{m} = \mathsf{CTS\text{-}Enc}_z(\mathsf{IV}, \mathsf{sec})$, and if $b = 1$, $\mathcal{C}$ executes H3, i.e. $\hat{m} \xleftarrow{\$} \{0,1\}^{253}$. In both cases, the generated signature σ, which contains $\hat{m}$ in its random numbers (c, r)'s, as shown in Fig. 3.6, is passed to $\mathcal{B}$, who is challenged to break the semantic security of CTS-Enc by distinguishing if the random numbers (r, s) are sampled randomly or encryptions generated by CTS-Enc. After receiving σ, $\mathcal{B}$ passes it to $\mathcal{W}$, who is challenged to distinguish whether σ was generated using H2 or H3. $\mathcal{W}$ outputs its guess as $b^* = 0$ to indicate σ was generated by H2, and $b^* = 1$ otherwise. After receiving $\mathcal{W}$'s output b^*, $\mathcal{B}$ outputs its guess b' based on b^*, particularly, $b' = b^*$. Since $\mathcal{W}$ distinguishes between H2 and H3 with a non-negligible probability, and $b' = b^*$, then $\mathcal{B}$'s advantage in breaking the semantic security of CTS-Enc, denoted by $\mathsf{Adv}_{\mathcal{B}}^{\mathsf{CTS\text{-}Enc}}$ is also non-negligible. *Hence, we conclude that H2 and H3 are computationally indistinguishable if* CTS-Enc *is a semantically-secure encryption algorithm.*

Since H1 and H2 are computationally indistinguishable if F is a secure PRF function, and H2 and H3 are indistinguishable if CTS-Enc is a semantically-secure encryption algorithm, then H1 and H3 are indistinguishable if F is a secure PRF and CTS-Enc is a semantically-secure encryption algorithm. Therefore, no PPT adversary $\mathcal{W}$ can win, with a non-negligible probability, the security game $\mathbf{Expt}_{\mathcal{W}}^{\mathsf{SU}}$ in Fig. 3.3 if F is a secure PRF and CTS-Enc is a semantically-secure encryption algorithm, which concludes our proof of Theorem 3.1. □

C.2 Security proof of Skywhisper (Theorem 4.1)

In the following, we use reduction to prove Theorem 4.1. Assume there exists a PPT adversary $\mathcal{A}$ who can break $\mathcal{ST}$ with a non-negligible advantage $\mathsf{Adv}_{\mathcal{A},\mathcal{ST}}^{\mathsf{CHA}}(1^\lambda)$ with respect to the CHA security experiment $\mathbf{Expt}_{\mathcal{A}}^{\mathsf{CHA}}(1^\lambda)$ in Fig. 4.1. We need to construct a PPT adversary $\mathcal{B}$ who can break the PRF game for Encrypt, i.e. $\mathcal{B}$ must be able to distinguish with a non-negligible probability between the pseudo-random text generated by the PRF and the randomly-selected text, as explained in Sec. 2.2.3.

During the reduction game, $\mathcal{B}$ plays as a challenger for $\mathcal{A}$ in the CHA game. Upon receiving m from $\mathcal{A}$, $\mathcal{B}$ picks random $\mathsf{rand} \xleftarrow{\$} \{0,1\}^{64}$ and sets $x = (\mathsf{rand} \| 00\ldots0)$. $\mathcal{B}$ then queries x to the PRF game challenger and obtains IV. Subsequently, $\mathcal{B}$ queries CT to the PRF game challenger, and obtains rand. $\mathcal{B}$ then computes (c, r) according to the description shown in Fig. 4.5 and Fig. 4.6. After that, $\mathcal{B}$ flips a coin $b \xleftarrow{\$} \{0,1\}$, and if $b = 0$, $\mathcal{B}$ computes a ring signature using (c, r); otherwise, $\mathcal{B}$ computes a ring signature normally. $\mathcal{B}$ then sends the resulting signature to $\mathcal{A}$, and $\mathcal{A}$ outputs a guess b^*. Assume the challenge bit in the PRF game is β, i.e. $\beta = 0$ in the PRF mode and $\beta = 1$ in the

random function mode. If $b = b^*$, $\mathcal{B}$ outputs $\beta^* = 0$; otherwise, $\mathcal{B}$ outputs $\beta^* = 1$.

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

Hence, the advtantage of $\mathcal{B}$ with respect to the PRF game is

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

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

C.3 Security proof of RRS

The security of RRS is defined by the advantage $\mathsf{AdvRRS}_{\mathcal{A}}$ of an adversary $\mathcal{A}$ in forging a randomized ring signature $\acute{\sigma}$. This advantage represents the adversary's probability in winning the following security game:

- **Setup:** The challenger generates $\{s_1, \ldots, s_n\} \leftarrow \mathbb{Z}_p^n$ and their respective public keys $PK_i = g_1^{s_i}$ for $1 \leq i \leq n$. $\mathcal{C}$ passes the generated public keys to $\mathcal{A}$.
- **Query:** $\mathcal{A}$ can request any ring signature on a message $m \in \{0,1\}^*$.
- **Output:** $\mathcal{A}$ generates a forged randomized ring signature $\acute{\sigma}_{\mathcal{A}}$.

$\mathcal{A}$ wins this game if $\acute{\sigma}_{\mathcal{A}}$ is a valid ring signature of the set on n public keys, and on a message m that has not been queried before. If there is not any PPT such adversary, then the RRS scheme is secure against forgeability.

Definition C.1. *A forger $\mathcal{A}(t, q, \rho)$-breaks the existential unforgeability EUF of the* RRS *scheme if $\mathcal{A}$ runs in time at most t, makes at most q queries, forges a randomized ring signature of n public keys, where $n > 1$, and his advantage $\geq \rho$. The scheme is (t, q, ρ)-secure and existentially unforgeable if no PPT adversary can (t, q, ρ)-break it.*

Theorem C.1. *The randomizable ring signature scheme* RRS *achieves the same security level as the* BGLSRingSignature *scheme. More precisely, if an adversary (t, q, ρ)-breaks the EUF of the* RRS *scheme, then there exists an adversary that can (t, q, ρ)-break the EUF security of the* BGLSRingSignature *scheme.*

Proof. Intuitively, a randomized ring signature $\acute{\sigma}$ is indistinguishable from the original signature σ to any third party: if $\mathsf{RingVerify}(L : \{PK_1, \ldots, PK_n\}, m \in \{0,1\}^*, \sigma)$ is valid then so is the verification of its respective randomized signature $\mathsf{RingVerify}(L : \{PK_1, \ldots, PK_n\}, m \in \{0,1\}^*, \acute{\sigma})$. Hence, if an adversary $\mathcal{A}$ forges a randomized ring signature $\acute{\sigma}$, another adversary $\mathcal{B}$ can pass it to the challenger $\mathcal{C}$ as a forged ring signature σ of the BGLSRingSignature scheme, without further processing or making any extra queries. □

For completeness, in the following, we explain Boneh et al.'s security proof of unforgeability of their BGLSRingSignature scheme [BGLS03]. They define the security of BGLSRingSignature as the advantage $\mathsf{AdvBGLS}_{\mathcal{A}}$ of an adversary $\mathcal{A}$ in the following security game:

- **Setup:** The challenger $\mathcal{C}$ generates a set of public keys $\{PK_1, \ldots, PK_n\}$ for N users, and these keys are passed to the adversary $\mathcal{A}$.
- **Hash Query:** $\mathcal{A}$ has unlimited access to a hash oracle, which he can query for the hash of any message m.
- **Sign Query:** $\mathcal{A}$ has unlimited access to a sign oracle, which he can query for the ring signature corresponding to any message m.
- **Output:** $\mathcal{A}$ outputs a forged ring signature $\sigma_{\mathcal{A}}$ using $\{PK_1, \ldots, PK_n\}$ on m, which has never been queried in during the Sign Query.

$\mathcal{A}$ wins this security game if the forged signature $\sigma_{\mathcal{A}}$ is a valid ring signature under the n public keys $\{PK_1, \ldots, PK_n\}$ on m that has not been queried before. If there is not any such PPT adversary, then the scheme is secure against existential forgeability.

Definition C.2. *An adversary $\mathcal{A}$ (t, ρ)-breaks the existential unforgeability EUF of the* BGLSRingSignature *scheme if $\mathcal{A}$ forges a valid ring signature under n public keys on a message m, runs in time at most t, and his advantage* $\mathsf{AdvBGLS}_{\mathcal{A}} \geq \rho$. *The scheme is (t, ρ)-secure and existentially unforgeable if no PPT adversary can (t, ρ)-break it.*

Assumption C.1. *The Computational co-Diffie-Hellman (co-CDH) problem is the following: given g_1, g_2 are generators for the groups $\mathbb{G}_1$ and $\mathbb{G}_2$, respectively, $g_2^a \in \mathbb{G}_2$, and $h \in \mathbb{G}_1$: compute $h^a \in \mathbb{G}_1$.*

Theorem C.2. *The* BGLSRingSignature *scheme achieves existential unforgeability under the co-CDH computational problem in Assumption C.1. Namely, if an adversary $\mathcal{A}$ can $(\acute{t}, \acute{\rho})$-break the existential unforgeability EUF of the* BGLSRingSignature *scheme, then there must exist another adversary who can (t, ρ)-solve the co-CDH problem. $t \leq 2\acute{t} + 2_{C_{G_2}}(2n + q_H + nq_S)$ and $\rho \geq ((\acute{\rho}/e)(1 + q_S))^2$, where $\mathcal{A}$ issues at most q_S queries to the sign oracle, and at most q_H queries to the hash oracle, and c_{G_2} is the time taken for exponentiation and inversion on $\mathbb{G}_2$.*

Proof. In the following, we explain Boneh et al.'s proof of their scheme and particularly Theorem C.2. Their proof shows that if there is an adversary $\mathcal{A}$ who forges a ring signature $\sigma_{\mathcal{A}}$, then there must exist another adversary $\mathcal{B}$ who solves the co-CDH problem. They construct $\mathcal{B}$ that given g_2^{ab}, g_1^a, and $a \neq 0$, computes g_2^b. This proof assumes that for every sign query on message m, $\mathcal{A}$ has previously requested a hash on the same message.

Setup: The challenger $\mathcal{C}$ generates $a, b \leftarrow \mathbb{Z}_p$ and passes g_2^{ab} and g_1^a to $\mathcal{B}$. The adversary $\mathcal{B}$ generates $\{x_2, \ldots, x_n\} \leftarrow \mathbb{Z}_p^{n-1}$ and sets $x_1 = 1$. It sets the public keys as follows: $PK_i = (g_1^a)^{x_i}$. After that, $\mathcal{B}$ passes $\{PK_i, \ldots, PK_n\}$ to $\mathcal{A}$. *Note that the private keys in this case consists of the set: $\{ax_1, ax_2, \ldots, ax_n\}$.*

Hash Query: When receiving a hash query from $\mathcal{A}$ on a message m_i, $\mathcal{B}$ flips a coin, which is 0 with probability p and 1 otherwise. Let us denote the coin toss result by c. $\mathcal{B}$ picks a random $z \xleftarrow{\$} \mathbb{Z}_p$. If c is 0, $\mathcal{B}$ sets $H(m_i) = (g_2^{ab})^z$, otherwise, it sets $H(m_i) = \psi(g_1^a)^z$. Here ψ is a computable isomorphism from $\mathbb{G}_1$ to $\mathbb{G}_1$. $\mathcal{B}$ returns $H(m_i)$ and stores the tuple $\langle m_i, c, z, H(m_i)\rangle$ in HashList.

Sign Query: When receiving a sign query from $\mathcal{A}$, $\mathcal{B}$ checks the HashList to find the coin toss result corresponding to the hash query on this same message. If $c == 0$, $\mathcal{B}$

fails and exists. Otherwise, $\mathcal{B}$ has returned $H(m_i) = \psi(g_1^a)^z$. $\mathcal{B}$ proceeds by generating a set of random scalars $\{y_2, \ldots, y_n\} \xleftarrow{\$} \mathbb{Z}_p^{n-1}$, computing $y_1 = z - (y_2 s_2 + \cdots + y_n s_n)$, and returning the ring signature $\sigma = \langle g_2^{y_1}, \ldots, g_2^{y_n} \rangle$.

Output: $\mathcal{A}$ outputs a forged ring signature $\sigma_{\mathcal{A}}$ on a message m. $\mathcal{B}$ checks the c value in HashList corresponding to the same message m. If c is 1, then $\mathcal{B}$ fails. Else, the corresponding hash value is $H(m_i) = (g_2^{ab})^z$, and $\mathcal{B}$ succeeds in solving the co-CDH problem, where g_2^b can be computed as the z^{th} root of $\sigma_1 \sigma_2^{x_2} \ldots \sigma_n^{x_n}$.

Note that $\mathcal{B}$ will not fail with probability $p^{q_S}(1-p)$, and it is also shown in [BGLS03] that this probability is maximum when $p = q_S/(q_S+1)$ giving a bound of $(1/e)(1+q_S)$. To calculate the running time of $\mathcal{B}$: it takes n exponentiations in $\mathbb{G}_1$ to generate the keys in the setup phase, one exponentiation for each hash query from $\mathcal{A}$, n exponentiations for each signature query from $\mathcal{A}$, and n exponentiations in the output phase, so $\mathcal{B}$'s running time is $\mathcal{A}$'s running time plus $c_{G_2}(2n + q_H + q_S)$. □

In the following, we illustrate the correctness of Boneh at al.'s proof by showing how $\mathcal{B}$ solves the co-CDH problem and successfully computes g_2^b. Since $\mathcal{A}$ is able to forge a valid ring signature $\sigma_{\mathcal{A}}$, then the signature element corresponding to the signer σ_j within $\sigma_{\mathcal{A}}$ is given as follows:

$$\sigma_j = \left(h / \psi(\prod_{i \neq j} PK_i^{r_i}) \right)^{1/a x_j}$$

Therefore, $\mathcal{B}$ can compute g_2^b as the z^{th} root of the following:

$$\begin{aligned}
\sigma_1 \sigma_2^{x_2} \ldots \sigma_n^{x_n} &= \prod_{i \neq j} \sigma_i^{x_i} \cdot \sigma_j^{x_j} = \prod_{i \neq j} g_2^{r_i x_i} \cdot (h/\psi(\prod_{i \neq j} PK_i^{r_i}))^{(1/a x_j) x_j} \\
&= \prod_{i \neq j} g_2^{r_i x_i} \cdot (h/\psi(\prod_{i \neq j} g_1^{(a x_i) r_i}))^{1/a} = \prod_{i \neq j} g_2^{r_i x_i} \cdot h^{1/a} / (\prod_{i \neq j} g_2^{x_i r_i}) = h^{1/a} \\
&= g_2^{abz/a} = g_2^{bz}
\end{aligned}$$

C.4 Proof of Dis-aggregation of BGLSAggregateSignature

In this section we prove that no PPT adversary can extract the signatures σ_i's from their corresponding aggregate signature σ_a when using the BGLSAggregateSignature scheme as explained in Sec. 6.2. We begin by explaining a three-phase dis-aggregation experiment between a challenger $\mathcal{C}$ who generates an aggregate signature and an adversary $\mathcal{A}$ who tries to extract at least one of the member signatures σ_i's that are part of σ_a.

- **Setup**: $\mathcal{C}$ generates n random scalars $\{s_1, \ldots, s_n\} \xleftarrow{\$} \mathbb{Z}_p^n$ and computes the corresponding public keys $\{PK_1, \ldots, PK_n\} \in \mathbb{G}_1$ where $PK_i = g_1^{s_i}$. $\mathcal{C}$ passes the n public keys to the adversary $\mathcal{A}$.
- **Query**: $\mathcal{A}$ queries $\mathcal{C}$ for an aggregate signature σ_a of the signatures on n messages $\{m_1, \ldots, m_n\} \in \{0,1\}^*$. $\mathcal{C}$ checks QueryList to find if any of the messages has been queried before, in which case the challenger $\mathcal{C}$ halts with failure. If $\mathcal{C}$ does not halt, it signs each message m_i using the corresponding secret key s_i to generate

σ_i. Then $\mathcal{C}$ generates the aggregate signature $\sigma_a = \prod_{i=1}^{n} \sigma_i$, $\sigma_a \in \mathbb{G}_2$. $\mathcal{C}$ stores the tuple $\langle \{m_1, \ldots, m_n\}, \{\sigma_1, \ldots, \sigma_n\}, \sigma_a \rangle$ in QueryList structure, and responds to $\mathcal{A}$ with σ_a.

- **Output**: finally, $\mathcal{A}$ outputs a single signature $\acute{\sigma_i}$ and wins the game if $\acute{\sigma_i} \subset$ QueryList.

Definition C.3. *An aggregate signature scheme achieves 'dis-aggregation' if there is not any probabilistic polynomial-time (PPT) adversary that can win the above dis-aggregation game with a non-negligible probability.*

Theorem C.3. *The* BGLSAggregateSignature *scheme achieves dis-aggregation under the co-CDH computational problem in Assumption C.1. If a PPT adversary $\mathcal{A}$ can win the dis-aggregation game with a non-negligible probability, then $\mathcal{A}$ can also solve the co-CDH problem with non-negligible probability.*

Proof. We prove Theorem C.3 using reduction to show that an adversary $\mathcal{A}$ who dis-aggregates a given aggregate signature σ_a to reconstruct at least one member signature σ_i, is also able to solve the underlying co-CDH problem. After the setup phase, $\mathcal{A}$ knows n public keys $\{PK_1, \ldots, \mathrm{PK}_n\}$, which are equal to $\{g_1^{s_1}, \ldots, g_n^{s_n}\}$. In this proof, $\mathcal{A}$ demonstrates his ability to solve the co-CDH by computing $H(m_i)^{s_i} \in \mathbb{G}_2$ given $g_1, g_1^{s_i}$, and $H(m_i)$. Note, we can assume that the messages m_i's are selected by another co-CDH challenger or sampled from a random oracle, but for simplicity, we let $\mathcal{A}$ choose the messages.

Following the same scenario of the above dis-aggregation game, in the output phase, the adversary $\mathcal{A}$ receives an aggregate signature σ_a. After that, $\mathcal{A}$ dis-aggregates σ_a to reveal at least one of the member signatures σ_i's. Having extracted at least one signature σ_j from σ_a, where $1 \leq j \leq n$, $\mathcal{A}$ has also solved the co-CDH problem since $\sigma_j = H(m_j)^{s_j}$. $\mathcal{A}$ can further check his solution by checking the following equality: $e(PK_j, H(m_j)) = e(g_1, \sigma_j)$. If this equality holds, then $\mathcal{A}$ has successfully solved the co-CDH problem. Also, since $\mathcal{A}$ dis-aggregates a given σ_a with a non-negligible probability, then $\mathcal{A}$ also solves the co-CDH problem with a non-negligible probability. □

C.5 Security Proof of RAS

In this section, we prove that the RAS scheme proposed in Sec. 6.3 is existentially unforgeable under chosen message attacks (EUF-CMA). As stated in Sec. 6.3, the first four algorithms of RAS are the same as Pointcheval and Sanders's randomizable signature, which was proved to be existentially unforgeable EUF-CMA under Assumption C.2 [PS16].

Assumption C.2. *Let $(p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e)$ a bilinear group setting of type 3, with g a generator for $\mathbb{G}_1$ and $\widetilde{g}$ a generator for $\mathbb{G}_2$. For $(\widetilde{A} = \widetilde{g}^a, \widetilde{B} = \widetilde{g}^b)$, where a and b are random scalars in $\mathbb{Z}_p$, we define oracle $\mathcal{O}(m)$ on input $m \in \mathbb{Z}_p$ that chooses a random $h \in \mathbb{G}_1$ and outputs the pair $\sigma = (h, h^{a+mb})$. Given $(\widetilde{g}, \widetilde{A}, \widetilde{B})$ and unlimited access to $\mathcal{O}$, no adversary can efficiently generate a valid σ, with $h \neq 1_{\mathbb{G}_1}$ for a new scalar m^* that has not been queried from $\mathcal{O}$.*

In the following, we refer to the first four algorithms of RAS as the 'single-message signature', and prove that the aggregability of our RAS scheme is secure against existential forgeability. The definition of the security of our aggregate signature is adapted from the

definition of the security of Boneh et al.'s aggregate signature [BGLS03]. The adversary $\mathcal{A}$ is given n public keys generated by the challenger $\mathcal{C}$. The adversary is allowed unlimited access to a signing oracle on any of the public keys. The advantage of the adversary $\mathsf{AdvAggSig}_{\mathcal{A}}$ is defined as the probability of winning the following security game, which consists of three phases: (1) setup, (2) queries, and (3) output.

Setup: The challenger $\mathcal{C}$ generates a set of n public keys $\{PK_1, \ldots, PK_n\}$, and passes them to the adversary $\mathcal{A}$, where $1 < n \leq N$ and N is a game parameter.
Query: The adversary can request signatures under any $PK_i \in \{PK_1, \ldots, PK_n\}$ on unlimited messages m.
Output: The adversary generates an aggregate signature σ_a on public keys $\{PK_1, \ldots, PK_n\}$ on messages $\{m_1, \ldots, m_n\}$.

The adversary wins the game if σ_a is a valid signature on $\{m_1, \ldots, m_n\}$ under $\{PK_1, \ldots, PK_n\}$, and $n > 1$. Also, it is required that one m_j corresponding to the public key PK_j has not been queried by $\mathcal{A}$ before, where $j \in \{1, \ldots, n\}$. If there is not any probabilistic polynomial time (PPT) adversary who can win this game with a non-negligible probability, then the aggregate signature is *existentially unforgeable under chosen message attacks* (EUF-CMA).

Definition C.4. *An aggregate forger $\mathcal{A}(t, q_G, N, \rho)$-breaks an N-user aggregate signature scheme in the aggregate chosen-key model if $\mathcal{A}$ runs in time at most t, makes at most q_G queries, the forged signature is by at most N users, and his advantage $\mathsf{AdvAggSig}_{\mathcal{A}} \geq \rho$. The aggregate signature scheme is (t, q_G, N, ρ)-secure and EUF-CMA if no PPT adversary can (t, q_G, N, ρ)-break it.*

Theorem C.4. *The aggregate signature scheme achieves the EUF-CMA security level under Assumption C.2. More precisely, if an adversary can break the EUF-CMA of the aggregate signature scheme with a non-negligible probability ρ, then there exists an adversary who breaks the EUF-CMA security of the single-message signature scheme with the same non-negligible probability ρ.*

Proof. In the following, we use reduction to prove Theorem C.4. We show that forging a valid aggregate signature σ_a is equivalent to successfully forging a valid single signature σ_c, which means breaking Assumption C.2. In order to do so, we describe a security game in which $\mathcal{A}$ is an adversary who forges an aggregate signature σ_a, $\mathcal{B}$ is another adversary trying to forge a single signature σ_j, and $\mathcal{C}$ is the challenger.

Setup: $\mathcal{C}$ generates a challenge public key $PK_c = (\widetilde{A}_c = \widetilde{g}^{a_c}, \widetilde{B}_c = \widetilde{g}^{b_c})$, and sends $(pp, \widetilde{g}, PK_c)$ to $\mathcal{B}$ where pp denotes public parameters. After receiving PK_c , $\mathcal{B}$ randomly generates a set of n public keys $\{PK_1, \ldots, PK_n\}$, and randomly picks a value j such that $j \xleftarrow{\$} [1, n]$. $\mathcal{B}$ sets the j^{th} public key as the challenge key, i.e. $PK_j = PK_c$, and stores the keys in $\mathsf{KeyList}$. Note that $\mathcal{B}$ knows the secret keys corresponding to all the public keys except for PK_j. After that $\mathcal{B}$ forwards $(pp, \widetilde{g}, \langle PK_1, \ldots, PK_n \rangle)$ to $\mathcal{A}$.
Query: $\mathcal{A}$ can request a signature from $\mathcal{B}$ on a message m under any public key $PK_i \in \{PK_1, \ldots, PK_n\}$. If $\mathcal{A}$ requests a signature under PK_j , $\mathcal{B}$ forwards the query to $\mathcal{C}$, otherwise $\mathcal{B}$ generates the signature itself by executing $\mathsf{Sign}(m, SK_i)$. In both cases, $\mathcal{B}$ responds to $\mathcal{A}$ with σ, and stores the tuple $\langle q_i, (\sigma_{q_i,1}, \sigma_{q_i,2}), m_{q_i}, PK_i, SK_i \rangle$ in a list $\mathsf{SignaturesList}$. q_i is a counter, which is used to allow to distinguish between the multiple queries that are made under the same public key.

Output: Eventually, $\mathcal{A}$ returns to $\mathcal{B}$ an aggregate signature σ_a on n messages $(m_1, \ldots, m_n)$ and using public keys $(PK_1, \ldots, PK_n)$, where $\mathcal{A}$ has queried $\mathcal{B}$ on all but a single message $m_i \in \{m_1, \ldots, m_n\}$. Importantly, $\mathcal{B}$ only proceeds if m_j that corresponds to the challenge key PK_j has not been queried by $\mathcal{A}$ before, i.e. $m_j \notin \mathsf{SignaturesList}$, otherwise $\mathcal{B}$ restarts the process. If m_j corresponding to the challenge key PK_j has not been queried by $\mathcal{A}$, then $\mathcal{B}$ considers σ_a a valid aggregate signature if it meets the following conditions:

- $\diamond$ $\mathsf{AggVerify}(\langle m_1, \ldots, m_n\rangle, \langle PK_1, \ldots, PK_n\rangle, \sigma_a) == 1$, i.e. the forged aggregate signature is accepted by $\mathsf{AggVerify}$ algorithm.
- $\diamond$ All public keys $(PK_1, \ldots, PK_n)$ are as stored in $\mathsf{KeyList}$.

If the two conditions are met, $\mathcal{B}$ parses the aggregate signature: $\sigma_a \overset{\mathsf{parse}}{\rightarrow} (\langle \sigma_{1,1}, \ldots, \sigma_{n,1}\rangle, \sigma_2)$. Next, using the stored $\sigma_{q_i,2}$'s in $\mathsf{SignaturesList}$, $\mathcal{B}$ extracts the ones that correspond to the signed messages, i.e. $\forall\ i \neq j : \sigma^*_{q_i} = \langle q_i, (\sigma_{q_i,1}, \sigma_{q_i,2}), m_{q_i}, PK_i, SK_i\rangle$ when $m_{q_i} == m_i$. After that, $\mathcal{B}$ computes the value of $\sigma_{c,2}$ using $\sigma_{j,2} = \sigma_2 \cdot (\prod_{i=1, i\neq j}^{n} \sigma_{q_i,2})^{-1}$. This reveals the signature corresponding to the challenge key PK_c: $\sigma_j = (\sigma_{j,1}, \sigma_{j,2})$. Since σ_a is a valid aggregate signature then:

$$e(\sigma_2, \widetilde{g}) = \prod_{i=1}^{n} e(\sigma_{i,1}, \widetilde{A}_i, \widetilde{B}_i^{m_i}) = e(\prod_{i=1}^{n} \sigma_{i,1}^{a_i + b_i \cdot m_i}, \widetilde{g}) = e(\prod_{i=1}^{n} \sigma_{i,2}, \widetilde{g})$$

Therefore, the forged single signature corresponding to the challenge key PK_c is $\sigma_j = (\sigma_{j,1}, \sigma_{j,2})$ is equivalent to $(\sigma_{j,1}, \sigma_{j,1}^{a_j + b_j \cdot m_j})$. This concludes our proof that if there is an adversary $\mathcal{A}$ who can break the EUF-CMA security of the aggregate signature and forge a valid σ_a, then there must exist an adversary $\mathcal{B}$ who can forge a single signature σ_c and consequently break Assumption C.2.

As j is randomly picked from $[1, n]$, the probability Pr_j that m_j has not been queried under the challenge key PK_j by $\mathcal{A}$ in the query phase, that is, the probability that $m_j \notin \mathsf{SignaturesList}$, is $1/n$, i.e. $\mathsf{Pr}_j = 1/n$. Therefore, on average, $\mathcal{B}$ will have to repeat the process n times. Hence, if the running time of $\mathcal{A}$ is t, then the running time of $\mathcal{B}$ is $n * t$. However, if $\mathcal{A}$ breaks the security of the aggregate signature with a non-negligible success probability of ρ, then there must exist another adversary $\mathcal{B}$ that can break the security of the single re-randomizable signature with the same non-negligible probability of ρ. This concludes our reduction security proof of Theorem C.4. $\square$