A measurement study of peer-to-peer bootstrapping and implementations of delay-based cryptography

Angelique Faye Loe

March 2024

Thesis submitted to Royal Holloway University of London for the degree of Doctor of Philosophy

Information Security Group
Department of Mathematics

Declaration

These doctoral studies were conducted under the supervision of Dr Elizabeth Anne Quaglia at Royal Holloway University of London.

This dissertation is the result of my own work whilst enrolled in the Information Security Group in the Department of Mathematics as a candidate for the degree of Doctor of Philosophy. Any results which were researched in collaboration with other stakeholders will be specifically indicated in the text. No part of this work has been submitted for any other degree or award in any other university or educational establishment.

Angelique Faye Loe
March 2024

Acknowledgements

I would first like to thank my supervisor, Elizabeth Anne Quaglia, for her unwavering guidance and support throughout my entire PhD journey. Her expertise, encouragement, and mentorship have been invaluable to me. She has supported me through three pregnancies, and I can attest she has worked beyond what her parental leave would entail. I am forever grateful for such a dedicated supervisor and friend.

I would next like to thank my co-authors, Liam Medley, and Christian O'Connell, for their dedication and insightful collaboration on our research projects.

I would also like to thank my examiners, Prof Liqun Chen, and Dr Maryam Mehrnezhad, and my chair, Prof Peter Komisarczuk. You all made my viva a wonderful life experience. You have all taken time out of your academic, professional, and personal lives to accept the invitation to review my research and conduct my face-to-face viva. Your willingness to contribute your expertise is genuinely appreciated.

I would also like to thank Dr Siaw-Lynn Ng and Prof Carlos Cid for their help in my annual reviews.

Finally, I would like to thank my family – Alice Loe, Rene Loe, Brian Loe, and Mark Foster – and my dearest friends, Toks Oladuti, Diana Oladuti, and their children Antonio, Xavier, and Natalya, for their support throughout the entire journey of my research. For all my challenges and successes, they had words of support and wisdom for each outcome.

Abstract

This thesis researches two distinct areas of study in both peer-to-peer networking for modern cryptocurrencies and implementations of delay-based cryptography.

The first part of the thesis researches elements of peer-to-peer network mechanisms, with a specific focus on the dependencies on centralised infrastructure required for the initial participation in such networks.

Cryptocurrencies rely on decentralised peer-to-peer networks, yet the method by which new peers initially join these networks, known as bootstrapping, presents a significant challenge. Our original research consists of a measurement study of 74 cryptocurrencies. Our study reveals a prevalent reliance on centralised infrastructure which leads to censorship-prone bootstrapping techniques leaving networks vulnerable to censorship and manipulation.

In response, we explore alternative bootstrapping methods seeking solutions less susceptible to censorship. However, our research demonstrates operational challenges and limitations which hinder their effectiveness, highlighting the complexity of achieving censorship-resistance in practice.

Furthermore, our global measurement study uncovers the details of cryptocurrency peer-to-peer networks, revealing instances outages and intentional protocol manipulation impacting bootstrapping operations. Through a volunteer network of probes deployed across 42 countries, we analyse network topology, exposing centralisation tendencies and unintentional peer exposure.

Our research also highlights the pervasive inheritance of legacy bootstrapping methods, perpetuating security vulnerabilities and censorship risks within cryptocurrency systems. These findings illuminate broader concerns surrounding decentralisation and censorship-resistance in distributed systems.

In conclusion, our study offers valuable insights into cryptocurrency bootstrapping techniques and their susceptibility to censorship, paving the way for future research and interventions to enhance the resilience and autonomy of peer-to-peer networks.

In the second part of the thesis, attention shifts towards delay-based cryptography, where the focus lies on the creation and practical implementations of timed-release encryption schemes. Drawing from the historical delay-based cryptographic protocols, this thesis presents two original research contributions.

The first is the creation of a new timed-release encryption scheme with a property termed implicit authentication. The second contribution is the development of a practical construction called TIDE (TIme Delayed Encryption) tailored for use in sealed-bid auctions.

Timed-Release Encryption with Implicit Authentication (TRE-IA) is a cryptographic primitive which presents a new property named implicit authentication (IA). This property ensures that only authorised parties, such as whistleblowers, can generate meaningful ciphertexts. By incorporating IA techniques into the encryption process, TRE-IA augments a new feature in standard timed-release encryption schemes by ensuring that only the party with the encryption key can create meaningful ciphertexts. This property ensures the authenticity of the party behind the sensitive data disclosure. Specifically, IA enables the encryption process to authenticate the identity of the whistleblower through the ciphertext. This property prevents malicious parties from generating ciphertexts that do not originate from legitimate sources. This ensures the integrity and authenticity of the encrypted data, safeguarding against potential leaks of information not vetted by the party performing the encryption.

TIDE introduces a new method for timed-release encryption in the context of sealed-bid auctions by creatively using classic number-theoretic techniques. By integrating RSA-OEAP public-key encryption and the Rivest Shamir Wagner time-lock assumption with classic number theory principles, TIDE offers a solution that is both conceptually straightforward and efficient to implement.

Our contributions in TIDE address the complexities and performance challenges inherent in current instantiations of timed-release encryption schemes. Our research output creates a practical timed-release encryption implementation on consumer-grade hardware which can facilitate real-world applications such as sealed-bid auctions with clear steps for implementation.

Finally, our thesis concludes with a review of the prospects of delay-based cryptography where we consider potential applications such as leveraging TIDE for a public randomness beacon.

Contents

List of Figures

List of Tables

List of Algorithms

Chapter 1

Introduction

This thesis explores two topics of research. The first part concentrates on peer-to-peer bootstrap mechanisms, with a specific focus on the dependencies on centralised infrastructure required for the initial participation in a peer-to-peer network. The second part focuses on the area of delay-based cryptography where we create two practical instantiations of timed-release encryption schemes.

1.1 Peer-to-Peer Network Research

In the first part of this thesis we research cryptocurrency peer-to-peer networks with a particular focus on the critical aspect of bootstrapping techniques. As the main focus of our research, we investigate how new peers on these networks achieve connectivity to join existing peer-to-peer networks.

To better frame the motivation for our research we first define the concept of trust. Trust is an important concept which motivates our area of research. We draw from the definition of trust provided by the Trusted Computing Group (TCG). The TCG is a consortium of industry-leading companies and organisations dedicated to developing and promoting open standards for trusted computing and security technologies. Established in 2003, TCG aims to enhance the security and integrity of computing systems through the development of hardware and software-based security solutions.

The TCG defines *trust* as: 'Trust is the expectation that a device will behave in a particular manner for a specific purpose [161].'

To make this definition meaningful to the content of Part 1 of our thesis we must substitute the word 'device' with the word 'system'.

Therefore, we adapt the TCG definition of trust to be: ‘Trust is the expectation that a system will behave in a particular manner for a specific purpose.’

This deviation allows us to consider a peer-to-peer network as a system, rather than the more restrictive definition of a device.

This definition is the basis for our examination of trust within the context of peer-to-peer cryptocurrency networks, where decentralisation is a key operational goal.

Cryptocurrencies, noted by prominent examples like Bitcoin [134] and Ethereum [54], espouse principles of decentralisation, ostensibly offering participants the promise of autonomy and freedom from centralised control. However, our investigation uncovers a challenge to the promise of decentralisation. Despite the ideological underpinnings of decentralisation, cryptocurrency networks often exhibit a significant reliance on centralised entities for crucial functions, notably in the area of network bootstrapping.

We were first motivated by the conjecture that cryptocurrency peer-to-peer networks were still dependent on trusted centralised resources for bootstrapping, rendering them vulnerable to similar weaknesses as their peer-to-peer file sharing predecessors. In our research we sought to review the discrepancy between the perceived decentralised behaviour of cryptocurrency networks and their actual dependencies on centralised elements for their basic operation. Our study had the motivation to explore the prevalent bootstrapping methods used by a diverse array of cryptocurrencies. We believed this would illuminate the pervasive use of heavily centralised and thus censorship-prone bootstrapping approaches.

As these networks profess to operate in a decentralised manner, the prevalence of centralisation in bootstrapping mechanisms challenges the very essence of how we trust these systems to behave in a particular manner for the specific purpose of being a decentralised peer-to-peer payment system. This contradiction between the purported decentralisation and the reliance on centralisation further motivated our research initiatives.

Our research motivation was not merely academic curiosity but also stemmed from a profound concern about the implications of this reliance on centralisation. By performing a measurement study on the bootstrapping techniques employed by contemporary cryptocurrencies, our aim was to provoke critical discourse and contribute to a deeper understanding of the centralised dependencies still inherent in contemporary cryptocurrency peer-to-peer networks.

While existing literature extensively covered research on peer-to-peer network bootstrapping [85, 117, 104] and the diverse methods aimed at reducing reliance on centralised resources [103], a notable research gap emerged in the specific context of cryptocurrency peer-to-peer networks. We identified a lack of comprehensive studies examining the likelihood of these networks relying on legacy bootstrapping methods, potentially leaving them open to censorship. Notably, there was a dearth of measurement studies investigating the aspect of peer-to-peer bootstrapping in the context of cryptocurrencies. This gap prompted our research initiative.

To address this research gap we conducted the first meticulous measurement study to assess the prevalence of legacy bootstrapping methods in cryptocurrency peer-to-peer networks. This study forms the key contribution to this part of our thesis. This measurement study contains contents from the following publication:

- You Shall Not Join: A Measurement Study of Cryptocurrency Peer-to-Peer Bootstrapping Techniques, from the Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security [116].

All contents and contributions to this publication were a result of my own original research under the supervision of Elizabeth Anne Quaglia.

1.2 Delay-based Cryptography Research

The second part of our thesis shifts its focus toward practical implementations of delay-based cryptography. Delay-based cryptography has emerged as an area of study, characterised by its approach to leveraging time as a security measure. Originating from the concept of time-lock puzzles introduced by Rivest, Shamir, and Wagner [151], delay-based cryptography has since evolved to more sophisticated timed-release encryption (TRE) schemes [62].

Modern timed-release encryption schemes aim to secure sensitive information by imposing temporal constraints on decryption, thus mitigating the risk of premature disclosure. Our research is motivated by the recognition of existing gaps in the literature regarding the implementation and practical applications of current timed-release encryption schemes. By addressing these gaps, we aim to contribute to this specific area of cryptographic primitives.

In this part of our thesis we collect the contents of two original research publications and partial elements of a systemisation of knowledge publication.

Our first original research paper focuses on the creation of a new property in timed-release encryption that we term implicit authentication (IA). Our motivation was to consider the safe disclosure of information by whistleblowers.

While conventional TRE schemes offer a degree of security through time-delay mechanisms [62], we saw that the current literature lacked certain safeguards against adversarial attempts to manipulate or impersonate the whistleblower's encrypted communications. This research gap prompted our exploration into the generation of a TRE scheme with the safeguarding property that we formally define as implicit authentication.

The introduction of (Timed-Release Encryption with Implicit Authentication) TRE-IA represents an augmented property to traditional timed-release encryption schemes, as it introduces a critical security property aimed at bolstering the integrity and authenticity of encrypted communications in sensitive contexts. Our practical construction extends the functionality of conventional TRE schemes and also addresses a need for enhanced an additional security measure for whistleblowers. By integrating implicit authentication into the current timed-release encryption, our aim to provide whistleblowers with another tool which offerts a predictable time-delayed decryption and assurances against unauthorised tampering or manipulation of their encrypted disclosures.

To address this research gap we created our TRE-IA delay-based cryptographic scheme. This study forms the one of the key contributions to this part of our thesis. This practical construction and implementation study contains contents from the following publication:

- Applications of Timed-Release Encryption with Implicit Authentication, from the Proceedings of AFRICACRYPT 2023, the 14th International Conference on Cryptology in Africa [113].

In the this paper I created the security games and the main construction, including all of the algorithms. The proofs were joint work with my co-authors Liam Medley and Elizabeth Quaglia, and the narrative and research of the introduction and background material was also joint work with my co-authors Liam Medley and Elizabeth Quaglia. The performance analysis was my joint work with Christian O'Connell and the Secure Drop integration contribution was another of my main contributions. This work was also completed under the supervision of Elizabeth Anne Quaglia.

In the second piece of research for Part 2 of our thesis, we focused on the area of timed-release encryption in the context of sealed-bid auctions. Building further upon the work by Chvojka et al. [62], which introduced the concept of TRE, we created a new practical construction called TIDE (TIme Delayed Encryption). Our motivation stemmed from the necessity to provide a scalable and practical solution to the complexities in Vickrey and general sealed-bid auctions.

The central challenge addressed by our research lay in devising a TRE scheme that ensured the delayed release of encrypted information but also facilitated the appropriate protocol of sealed-bid auctions. Traditional approaches to the use of delay-based cryptography for sealed-bid auctions often relied on cryptographic primitives such as commit-and-reveal protocols or homomorphic encryption, which may have suffered from scalability or efficiency issues [43, 120, 53]. Our research sought to bridge these issues by presenting TIDE, a practical construction specifically designed for use in sealed-bid auctions. By combining RSA-based public-key encryption with time-lock puzzles in a novel and practical way, our goal was to create a practical implementation that was also performance-tested.

The underlying construction principles remained rooted in a similar cryptographic framework as TRE-IA. However, using different number theoretic techniques, we shifted our focus towards addressing the challenges inherent in Vickrey and other forms of sealed-bid auctions. Through a detailed security analysis and an implementation study, our goal was to demonstrate the efficacy and feasibility of TIDE.

To address some of the practical challenges with current delay-based cryptographic schemes in the context of sealed-bid auctions we created our TIDE scheme. This study forms the another one of the key contributions to this part of our thesis. This practical construction and implementation study contains contents from the following publication:

- TIDE: A Novel Approach to Constructing Timed-Release Encryption, from the Proceedings of the ACISP 2022, the Australasian Conference on Information Security and Privacy [112].

In this paper I also created the security games and the main construction including all of the algorithms. The proofs were joint work with my co-authors Liam Medley and Elizabeth Quaglia, and the narrative and research of the introduction and background material was also joint work with my co-authors Liam Medley and Elizabeth Quaglia. The practicality and implementation was my joint work with Christian O'Connell, where Christian was mainly involved in the creation of the Figures and the setup of the Raspberry Pi devices. This work was also completed under the supervision of Elizabeth Anne

Quaglia.

Finally, we note the work completed in the first chapter of Part 2 of our thesis. In this chapter we provide the background information regarding the history and development of delay-based cryptography to help shape the context of our original research. This chapter also contains all of the neccessary number theory used in our original research publications for TRE-IA and TIDE.

It is important to highlight the rationale behind placing the discussion on the systemisation of knowledge at the end of this introduction to delay-based cryptography research. While this work represents a minor contribution to this part of the thesis, it serves a crucial purpose in consolidating the vast array of literature within the field.

As part of our background research into the area of delay-based cryptography we identified many sources of literature often with divergent definitions and subtle implementation variances. We saw an opportunity near the end of our research journey to cullminate our knowledge in a systemisation of knowledge of delay-based cryptography.

To address the variances and differences in various delay-based cryptographic literature we published a systemisation of knowledge paper. This work contains contents from the following publication:

- SoK: Delay-Based Cryptography, from the Proceedings of the 36th IEEE Computer Security Foundations Symposium 2023 [114].

In this paper, Liam Medley assumed the principal authorship. I authored the segment encompassing timed commitments and timed signatures, and additionally made contributions to sections covering time-lock puzzles, time-lock encryption, timed-release encryption, and delay encryption. This work was completed under the supervision of Elizabeth Anne Quaglia.

It is noteworthy that only a modest fraction of this paper is incorporated into my thesis. As stated, partial contents of this systemisation of knowledge serve as some of the contents of the bridging chapter to establish the context for my primary research in the first and second papers, respectively.

Part 1: Centralised Dependence in Peer-to-Peer Networks

Part 1 of this thesis begins with Chapter 2 entitled ‘Background of Peer-to-Peer Networks’ which explores the history of peer-to-peer networks. This chapter sets the background and context of why the reliance on centralised resources in peer-to-peer networks used for file-sharing led to their censorship and decline.

Part 1 of this thesis then presents the main body of research with Chapter 3 entitled ‘A Measurement Study of Cryptocurrency Peer-to-Peer Bootstrapping Techniques’ which provides our detailed measurement study of cryptocurrency bootstrapping methods. This chapter provides insight into the enduring prevalence of centralised dependencies for modern cryptocurrency peer-to-peer networks.

This chapter challenges the level of trust individuals investing in and trading digital assets place in the presumed decentralisation of cryptocurrencies. However, despite the anticipated decentralised behaviour of these systems, our analysis uncovers an inherent reliance on centralised resources for critical functionalities.

Chapter 2

Background of Peer-to-Peer Networks

In this chapter we explore the origins of early peer-to-peer networks and provide context into their historical evolution. Subsequently, we examine the pivotal role that the dependence on central resources plays in the basic functionality of these systems. We show how this dependence on centralised resources eventually led to the decline of these early peer-to-peer systems.

Next, we provide the motivation underpinning our decision to research the sustained reliance on centralised third parties in modern peer-to-peer networks. Peer-to-peer networks represent a decentralised approach to computing and data sharing, where participants (peers) share resources, services, and information directly with each other, rather than through a central server. Peer-to-peer networks have revolutionised various aspects of digital communication, from file sharing to cryptocurrency transactions. The origins of these networks can be traced back to the motivation of decentralising control and enabling efficient resource sharing among users.

The concept of peer-to-peer networks emerged as a response to the limitations of traditional client-server architectures. In the early days of the internet, most communication relied on central servers, which posed challenges in terms of scalability, fault tolerance, and single points of failure. Peer-to-peer networks aimed to address these issues by distributing tasks and resources across a network of peers.

One of the earliest and most influential motivations for peer-to-peer networks was efficient and decentralised file sharing. These networks democratised content distribution, enabling users to share and access files without relying on centralised servers. In the next sections we provide a historical view of peer-to-peer file sharing networks and discuss the role of centralised resources in these systems.

2.1 Centralised Dependency for Peer-to-Peer Bootstrapping

In a peer-to-peer network, when a new node wishes to join the existing network of peers it begins a process known as bootstrapping. Typically, this begins by connecting to one or more predetermined nodes. These initial connections enable the newcomer to gather information about other nodes and their addresses, facilitating the expansion of its network participation. As these connections to other peers are established, the new node progressively integrates into the peer-to-peer ecosystem, contributing to the decentralised and collaborative nature of the network.

In Figure 2.1 we demonstrate the centralised dependence that new peers on a peer-to-peer network rely on to connect to an existing established network. The figure demonstrates that dependence in a centralised resource is commonly required for bootstrapping as it provides an initial set of reliable nodes or network addresses that newcomers can use to start their connections. In the next section we describe how the reliance on these centralised resources for bootstrapping has played a significant role in the downfall and censorship of early peer-to-peer networks.

2.2 Historical Peer-to-Peer File Sharing

In the late 1990s and early 2000s the first wave of peer-to-peer technologies became mainstream in the form of file sharing networks. We recall some legal history surrounding peer-to-peer file sharing networks such as Napster, Gnutella, Limewire, KaZaa, BitTorrent, and The Pirate Bay (TPB) [86]. Following the rapid growth of these peer-to-peer file sharing networks, trade organisations including the Recording Industry Association of America (RIAA) and the Motion Picture Association of America (MPAA) began several legal campaigns to shut down these systems citing copyright infringement laws, such as the Digital Millennium Copyright Act.

Focusing our attention on the two most prominent file sharing platforms, Napster and The Pirate Bay, we document how the reliance on centralised resources assumed a crucial role in the decline of these services. The legal accountability pertaining to copyright infringement was efficiently addressed technologically by the MPAA and RIAA. The

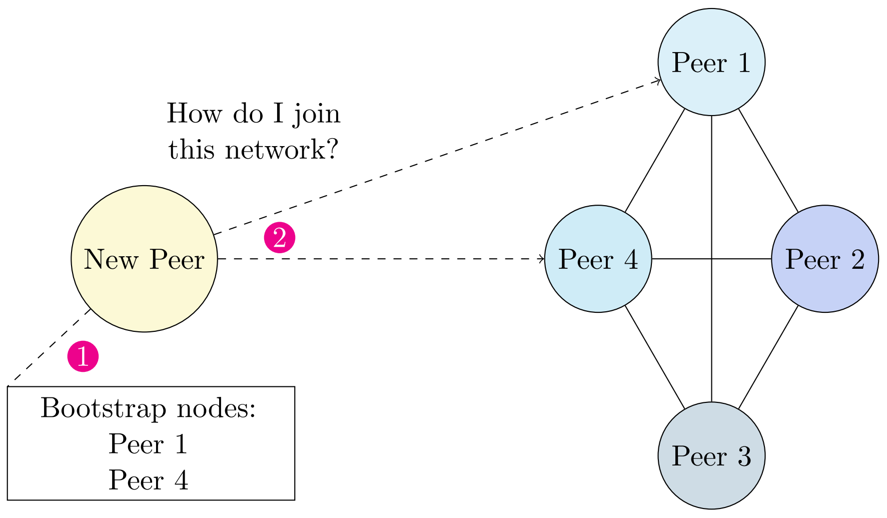

Figure 2.1: Peer-to-Peer Bootstrapping – A new peer must learn how to join an existing peer-to-peer network. Step 1 shows the centralised list of bootstrap nodes that the new peer must be informed of to connect and Step 2 shows the subsequent connection to these nodes to join the network.

case of Napster serves as a poignant example, having led to the complete cessation of its network operations. Similarly, in the instance of The Pirate Bay, we briefly explore the prevailing blocking mechanisms that have been instituted to curtail peer engagement due to the dependence on centralised resources.

2.2.1 Napster shut down

Napster was forced to shut down its peer-to-peer file sharing service in July 2001 after losing its legal battle with the RIAA. In a 2013 interview with Fortune magazine, a former executive recalled explicitly how the Napster service was ultimately shut down due to a dependency on centralised front-end servers which allowed peers to learn about the content hosted on other peers on the network [136]. We see that despite the peer-to-peer nature of the Napster network, peers would need to initially rendezvous to centralised servers in order to find the files they sought hosted on various peers. The dependency on these centralised servers facilitated the ease at which the file sharing service could be shut down. In retrospect, the Napster network was a hybrid peer-to-peer and client-server model, due to its dependency on centralised servers for client rendezvous.

This reliance on centralised servers for initial peer rendezvous demonstrated the dependence on centralised entities for facilitating connections and discovering shared content. However, this dependence also made the file sharing service vulnerable, as the shutdown of these centralised servers led to the network's downfall.

2.2.2 The Pirate Bay

The Pirate Bay (TPB) was founded in 2003. TPB, like Napster operates in a hybrid peer-to-peer and client-server model. However, unlike Napster, it is presently still in operation. Depending on which country access originated from there are various layers of Internet censorship limiting access to TPB.

Throughout its existence The Pirate Bay has been flexible with its infrastructure design and protocol utilisation in anticipation of legal battles and subsequent shut down orders[1]. An example of this flexibility can be noted by the original use of centralised *tracker servers* that provided the rendezvous point for peers to find the relevant torrent files. In this way, TPB had an infrastructure model that mirrored that of its defunct predecessor Napster. However, after various law enforcement episodes, in 2012, TPB opted to move away from the use of torrent files on these centralised servers in favour of magnet links [89] and distributed hash tables [101]. It also moved to flexible and geographically diverse cloud infrastructure for their server hosting [70].

These infrastructure and protocol changes were put in place to mitigate the dependency on elements of their centralised architecture, thereby making the tasks of blocking and potential shut down for their opponents more challenging. However, despite these mitigation techniques, blocking mechanisms currently used to limit access to The Pirate Bay are still relatively effective due to the underlying dependence placed on centralised resources. The two techniques commonly employed by ISPs (Internet Service Providers) to restrict access to file sharing sites include IP based deny-lists and DNS level blocking [118].

In Chapter 3 our research shows that the blocking mechanisms that are currently executed by ISPs under legal obligation to uphold copyright infringement laws are also effective at blocking modern peer-to-peer networks due to their dependence on the same centralised services.

[1]A detailed review covering the history of The Pirate Bay and its various legal battles, shut downs and resurrections can be found in [155, 71].

2.3 Modern Peer-to-Peer Networks

The evolution of peer-to-peer networks extended beyond file sharing to other domains, including the realm of cryptocurrencies. Cryptocurrencies are digital, decentralised currencies that utilise peer-to-peer networks to enable secure transactions without the need for intermediaries like banks. They often employ consensus algorithms to validate transactions and maintain the network's integrity.

In the context of cryptocurrencies, our research demonstrates that the dependence on centralised resources is still required to join a peer-to-peer network. The dependence on a centralised resource to join a peer-to-peer network conflicts with the motivation for reduced reliance on a centralised party to obtain consensus in financial transaction execution, clearing, and settlement. Cryptocurrencies use consensus algorithms like Proof of Work (PoW) or Proof of Stake (PoS) to validate and record transactions on a distributed ledger [2]. These mechanisms replace the need for a central authority to ensure the network's security and integrity.

Both peer-to-peer file sharing and cryptocurrencies began as disruptive pioneering technologies which experienced a rapid growth phase into mainstream adoption. In Table 2.1 we provide a brief comparison of the two technologies. The table shows that Napster was the 'case zero' technology in use for peer-to-peer file sharing and Bitcoin was the first dominant cryptocurrency to be adopted. The table shows a sample of the second-generation 'spin-off' technologies which proliferated after the pioneering technology was established. The table also highlights the opposition to the expansion and widespread adoption of the technologies. For file sharing networks, they began to attract opposition because of copyright infringement laws, and similarly, we see that cryptocurrencies have come under the scrutiny of financial and environmental regulators [11, 23]. For both file sharing and cryptocurrencies, their critics and challengers have sought to control and regulate elements of their systems in response to the undesirable disruption caused by their operation.

By highlighting the similarities to peer-to-peer file sharing we see evidence of a comparable narrative for cryptocurrencies emerging. This compelling narrative provides critical insight into the current expansion of regulation towards cryptocurrencies. Regu-

[2]Although consensus algorithms, such as the Proof of Work used in Bitcoin, seek to minimize the reliance on centralised third parties such as banks for financial transactions, there is a growing debate about their growing energy demands and the environmental implications this carries [69, 127]. Fortunately, there is a growing body of research dedicated to mitigating the energy demands associated with this disruptive technology [115, 58].

Table 2.1: Peer-to-Peer Technologies: File Sharing vs. Cryptocurrencies

peer-to-peer realm	**Main Years**	**Pioneer**	**Spinoffs**	**Legal Challenges**	**Challengers**
File Sharing	1999 - Present	Napster	Limewire, eDonkey, Gnutella, TPB	copyright violation	RIAA, MPAA
Cryptocurrencies	2009 - Present	Bitcoin	Litecoin, Ethereum, Dash, Monero	financial regulation, energy regulation	SEC, CFTC

The Security and Exchange Commission (SEC), and the Commodity Futures Trading Commission (CFTC) are US government agencies.

lation curtailing unrestricted access to cryptocurrencies could be established by focusing on the centralised components that necessitate fundamental aspects required for their basic operation.

Chapter 3

A Measurement Study of Cryptocurrency Peer-to-Peer Bootstrapping Techniques

Cryptocurrencies are digital assets which depend upon the use of distributed peer-to-peer networks. The method a new peer uses to initially join a peer-to-peer network is known as bootstrapping. The ability to bootstrap without the use of a centralised resource is an unresolved challenge.

In this chapter we survey the bootstrapping techniques used by 74 cryptocurrencies and find that censorship-prone methods such as DNS seeding and IP hard-coding are the most prevalent. In response to this finding, we tested two other bootstrapping techniques less susceptible to censorship, Tor, and ZMap, to determined if they were operationally feasible alternatives more resilient to censorship.

We perform a global measurement study of DNS query responses for each the 92 DNS seeds discovered across 42 countries using the distributed RIPE Atlas network. This provides details of each cryptocurrencies' peer-to-peer network topology and also highlights instances of DNS outages and query manipulation impacting the bootstrapping process. Our study also reveals that the source code of the cryptocurrencies researched comes from only five main repositories; hence accounting for the inheritance of legacy bootstrapping methods. Finally, we discuss the implications of our findings and provide recommendations to mitigate the risks exposed.

This chapter appears as part of the Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security, November 2019, on Pages 2231–2247, with the title '*You Shall Not Join: A Measurement Study of Cryptocurrency Peer-to-Peer Bootstrapping Techniques*' [116]. All work was completed under the supervision of Elizabeth Anne Quaglia.

3.1 Introduction

Cryptocurrencies are decentralised digital assets that do not rely on centralised third parties for the execution of transactions between parties. Decentralisation is achieved through the use of distributed ledger technology to provide an append only chronicle of all transactions between parties sending and receiving funds. The immutability of transactions is achieved through a fiscally incentivised consensus protocol publicly executed by peers participating in a peer-to-peer network [134]. The decentralised nature of cryptocurrencies provides a promise of technological amelioration to traditional payment systems [134, 73, 163, 38]. Due to the unprecedented pace of mainstream adoption and investment into these digital assets, they have attracted considerable scrutiny and research [26, 129, 95, 41, 107, 128, 72, 92, 95, 131, 133]. We now present the contributions of our measurement study of the techniques used to initially connect 74 different cryptocurrency peers to their relevant distributed peer-to-peer networks.

3.1.1 Contributions

When a new peer first joins a peer-to-peer network the action describing this initial connection is known as *bootstrapping*. By surveying the bootstrapping techniques of these cryptocurrencies we uncover several elements about their operation. Firstly, we note their inability to resist known censorship techniques to prevent peer connectivity. Secondly, we review the feasibility of employing censorship-resistant connection techniques. Next, we provide details of their peer-to-peer traffic and network infrastructures. Next, we present evidence of connection manipulation and outages impacting peer connectivity. Finally, we give insight into the inheritance of their source code which identifies the reasons for their use of censorship-prone bootstrapping techniques. Based on our findings we were able to present a number of security and social implications of cryptocurrency bootstrapping censorship not known from previous studies [21, 162].

Our research also contributes to the broader understanding of trust within decentralised systems. By considering the Trusted Computing Group's (TCG) definition of trust with the operational reality of cryptocurrency bootstrapping, we provide valuable insights into the discrepancy between perceived decentralisation and actual centralised dependencies. This research will aid individuals to be able to critically assess the trust they place in cryptocurrencies, prompting a reevaluation of their decentralised nature.

3.1.2 Overview of Measurement Study

We first review the bootstrapping techniques known from related studies into distributed system connectivity. We classify the techniques into three categories based on their ability to withstand censorship and record the advantages and disadvantages of each method. The categories identified are:

- censorship-prone methods, such as Domain Name System (DNS) seeding and Internet Protocol (IP) hard-coding,
- censorship-mitigated methods, such as via Internet Relay Chat (IRC) or The Onion Router (Tor),
- censorship-resistant techniques such as IPv4 scans using ZMap [85, 16, 22].

In the context of centralisation, censorship-prone methods require the highest degree of reliance on centralised resources, censorship-mitigated methods begin to reduce the amount of dependance required on centralised resources, and censorship-resistant techniques require the least amount of dependance on centralised resources. Due to the widespread use of cryptocurrency exchanges, we also identify how these centralised third parties fit into the peer-to-peer ecosystem in relation to bootstrapping functionality.

Next, we present the results of our survey to determine the bootstrapping techniques used by the cryptocurrencies researched. The survey highlights that 95% of cryptocurrencies use censorship-prone techniques of bootstrapping based on either DNS seeding or IP hard-coding. Furthermore, 32% of cryptocurrencies use a single DNS provider for their DNS seeds, exposing a single point of service failure. Also, 88% of the cryptocurrencies use distinctive destination ports, making traffic identification trivial. Based on these findings we employ known censorship techniques to ascertain if any fallback techniques of bootstrapping exist on the cryptocurrencies tested. Unfortunately, in all but one case our findings highlight that fallback techniques were equally prone to basic censorship. This led us to test the feasibility of bootstrapping via a censorship-mitigated technique using Tor, which presented various results for connectivity success. Finally, we test the ability to bootstrap via a censorship-resistant technique using a peer initiated ZMap IPv4 scan. Unfortunately, this technique is limited by large latency overheads and, most importantly, not a single cryptocurrency could connect using this method.

In the second part of our measurement study, as a consequence of highlighting the prevalence of DNS seeding as a bootstrapping method, we exploit this side-channel to measure details about the cryptocurrencies peer-to-peer networks over a number of months. Using the globally distributed network of RIPE Atlas probes, our research consists of measuring the details of the DNS query responses to the 92 DNS seeds uncovered during our bootstrapping survey. A RIPE (Réseaux IP Européens) Atlas probe is a small hardware device deployed by volunteers to measure and monitor various aspects of internet connectivity and performance, contributing to a global network measurement infrastructure.

This study is done across 46 locations in 42 different countries with varying legal standpoints towards cryptocurrencies. Our measurements uncover the topology of each cryptocurrencies' peer-to-peer network and highlight that some cryptocurrencies use centralised rendezvous servers as part of their connectivity strategy whilst others unintentionally expose the IPs of their peers through the harvesting of DNS query responses. Through this measurement study we also determine a range of statistics regarding the number of peer IPs seen on a per cryptocurrency and per country basis, as well as highlighting the number of IPs active across multiple cryptocurrencies. Our data also reveals how DNS seeding outages and query response manipulations impact bootstrapping operations in 60% of the countries we investigate.

Furthermore, our research exposes that the root cause of the prevalence of censorship-prone bootstrapping techniques is due to the widespread practice of copying source code. Our findings highlight that all of the cryptocurrencies researched derive from only five parent source code repositories. Additionally, we present a number of security implications and social implications related to cryptocurrency censorship based on the discoveries in our study. We conclude by providing a catalogue of tactical and strategic recommendations to mitigate the shortcomings to cryptocurrency bootstrapping identified by our research.

3.2 Background and Related Work

In this section we examine the details of peer-to-peer bootstrapping. We then perform an analysis of related work outlining various peer-to-peer bootstrapping techniques and classify them based on their ability to withstand censorship. Finally, we discuss how cryptocurrency exchanges fit into the process of peer-to-peer bootstrapping to facilitate the trade and acquisition of cryptocurrencies.

3.2.1 Peer-to-Peer Bootstrapping

The method in which new peers initially join a peer-to-peer network is commonly referred to as bootstrapping. Bootstrapping can be succinctly described as '*the process that a new peer who intends to join a peer-to-peer network uses to discover contact information for another peer in the existing network*' [85]. The ability to bootstrap onto a peer-to-peer overlay network without the use of centralised resources remains an outstanding challenge [117]. Any centralised elements on a peer-to-peer network mark a point for regulation and censorship and are ideally avoided.

However, a challenge arises when a new peer with no knowledge of other peers wishes to join the network. They must first be able to determine if they are the first peer on the network and, if not, they must be able to find and connect to at least one other peer in order to join the existing peer-to-peer network [103, 104]. We will now review the methods of peer-to-peer bootstrapping.

3.2.2 Censorship-Prone Bootstrapping

In this section we review two legacy methods of peer-to-peer bootstrapping which are heavily prone to censorship. For context, we also provide two brief examples of peer-to-peer networks impacted by bootstrapping censorship.

The first method to bootstrap peers onto a peer-to-peer network is to ship the software with a preconfigured list of peer IP addresses, known as hard-coding. Hard-coding is not ideal because the list of peers can quickly become obsolete and it gives an adversary a method to learn the details about the peer-to-peer network [85].

The second censorship-prone method to bootstrap onto a peer-to-peer network is the use of DNS seeds. DNS is a hierarchical client-server protocol which maps domain names to IP addresses [110]. For example, a new peer querying a DNS server for the DNS seed `node-london.cryptonex.org` for the Cryptonex cryptocurrency will have an A record return a single IP or multiple IPs. An A record is a type of DNS record that maps a domain name to an IPv4 address, enabling the translation of human-readable domain names into numerical IP addresses for internet communication [110]. The peer will then try to connect to the IPs when bootstrapping onto the network. DNS seeding is not ideal because DNS is a client-server based Internet protocol which is easily censored due to its centralised nature and mainly cleartext communication mechanisms.

Both methods have the advantage of being easy to configure and implement. The developer need only hard-code the relevant IPs or DNS seeds into the source code. DNS seeding also requires the additional step of configuring the DNS zone and records. We will also see in Section 3.5.2 that DNS seeding unintentionally leaks information about peer IP addresses.

The ability to censor peer-to-peer networks based on these bootstrapping methods can be recalled from peer-to-peer file-sharing technologies used to exchange digitally encoded music and videos in the late 1990s and early 2000s [86]. The two most prominent file sharing services, Napster and The Pirate Bay (TPB), were shutdown or censored after they were found legally accountable for copyright infringement. Napster was shutdown in 2001 after losing a legal battle with the Recording Industry Association of America. The service was easily shutdown due to the use of a centralised bootstrapping method dependent on front-end servers which allowed peers to learn about the content hosted on other peers on the network [136].

Censorship mechanisms currently used to limit access to TPB are IP deny-lists and DNS level blocking, both of which would be highly effective at censoring IP hard-coding and DNS seed bootstrapping methods [118].

IP deny-lists are simple file-based lists which ensure that access to specific resources on particular IP addresses are blocked [139]. They are typically implemented on network devices such as firewalls. A common method to block DNS is a technique known as DNS sinkholing. DNS sinkholing involves redirecting domain name resolution requests to a specific server which was not the originally intended server which effectively blocks the access [52].

3.2.3 Censorship-Mitigated Bootstrapping

In this section we explore the related work covering two bootstrapping methods which address the disadvantages of IP hard-coding and DNS seeding.

The first method relies on the use of IRC (Internet Relay Chat) servers [103]. IRC is a client-server Internet protocol which allows clients to send chat messages to other clients via distributed chat servers. This method exploits the distributed network of IRC servers as rendezvous points for bootstrapping. By using these servers, centralised points of failure for initial peer connection are removed. The issue with this method is

Table 3.1: Comparison of Peer-to-Peer Bootstrapping Methods. Noted columns: vulnerability to censorship, ease of configuration, if hard-coding is shifted to another method, such as IRC (Internet Relay Chat) or Tor, if the other shifted method is also vulnerable to censorship, centralised dependencies, obfuscating peer-to-peer traffic, high latency and high bandwidth requirements. ● = yes, ○ = no, × = N/A.

Method	***Censorship***	***ease of conf***	***shift h/c***	***method cens.***	***central dep.***	***hides P2P***	***high latency***	***high b.w.***
Hard-Coding	prone	●	×	×	○	○	○	○
DNS Seeding	prone	●	×	×	●	○	○	○
via IRC	mitigated	○	●	●	●	○	○	○
via Tor	mitigated	○	●	●	○	●	●	○
ZMap Scan	resilient	○	○	○	○	○	○	●

the underlying requirement to find an IRC server to connect to. Unfortunately, finding IRC servers also depends on using a hard-coded list of IPs or networks, or via DNS seeds [14]. Due to this deficiency, IRC bootstrapping is not widely deployed.

The second censorship-mitigated bootstrapping technique is dependent on the use of Tor hidden services. This method channels peer-to-peer traffic via the Tor network. Tor is a volunteer-based overlay network enabling its users to browse the Internet anonymously [16]. To obtain anonymity, traffic is first encrypted and then passed through a series of at least three hops, namely an entry, middle, and exit relay. An advantage of this method is that peer-to-peer traffic appears to be Tor traffic. Unfortunately, several countries wishing to regulate and censor Internet access attempt to detect and block Tor and other VPN (Virtual Private Network) traffic [105]. Censoring Tor is possible because default configurations of Tor are easily identified based on distinct TCP destination port usage and the characteristics of the TLS (Transport Layer Security) handshake between the Tor client and entry relay [3]. Tor can also be censored at IP level because all Tor relays can be identified by publicly available information stored in Tor directory authorities.

Tor addresses these shortcomings by introducing *pluggable transports* and Tor bridges [122]. Pluggable transports utilise the steganographic concept of *security by obscurity* by attempting to make Tor traffic appear as standard TLS traffic. In this way, pluggable transports provide an advantage as peer-to-peer traffic is more difficult to identify.

Tor bridges are entry relays which do not have their IPs publicly available on Tor directory authorities [122]. Unfortunately, default Tor bridges also suffer from the dependence of hard-coding IPs into the Tor browser bundle leaving them vulnerable to censorship. However, to help circumvent censorship private Tor bridges are not hard-coded into the Tor browser bundle. By design, the discovery of private bridges depends on the use of manual side-channel requests, such as email. This discovery overhead makes the use of private bridges for peer-to-peer connectivity challenging.

One of the biggest disadvantages of utilizing Tor for peer-to-peer bootstrapping is the bandwidth limitations and latency. This is due to the overheads of the three-hop encrypted relay design. In the case of bootstrapping peer-to-peer cryptocurrencies with a requirement to download large blockchains, the former shortcomings can pose significant operational issues, as we shall see in Section 3.4.3. Finally, it is important to note that other research observes that Tor is not a panacea for anonymity, especially when considering cryptocurrencies [41, 90].

3.2.4 Censorship-Resistant Bootstrapping

We finally discuss related work regarding the most censorship-resistant method of bootstrapping based on Internet Protocol version 4 (IPv4) [142] scanning.

The earliest peer-to-peer bootstrapping scan-based method was based on geographically targeted IP scanning [85]. The idea behind this method is the creation of a profile of the IPs already part of the network. Once the IPs are determined, a targeted scan is formulated based on distribution information learnt from DNS. The advantage of this method is the focus to remove centralised elements of bootstrapping and the need to hard-code IPs. Unfortunately, this method still relies on the centralised DNS protocol and also assumes that the peer-to-peer network is already established so that learning the details of the peer IPs is possible.

However, this method importantly explores a peer-initiated scan of the IPv4 address space in order to learn about other peers. The computational feasibility to scan the entire IPv4 address space is now possible due to advances in the ZMap network scanner [22] [1]. ZMap is able to scan the entire IPv4 address space for as single Transmission Control Protocol (TCP) [143] port in 4.5 minutes given a 10 Gpbs Ethernet connection. The advantages of this scanning method is the removal of any centralised dependency.

[1]It is not feasible for ZMap to scan the entire IPv6 address space. The IPv6 address space contains nearly 2^{128} unique addresses. If it were feasible to scan the entire IPv6 address space, it would also be feasible to perform a search on the (Advanced Encryption Standard) AES-128 key space.

However, the stated scan time of 4.5 minutes assumes that a high bandwidth connection to the Internet is available. Also, this technique of bootstrapping is very bandwidth intensive. It generates 4.6 Gbps of traffic for each peer wishing to bootstrap [22]. Furthermore, typical broadband speeds are below 10 Gbps. Broadband speeds in Singapore average 60 Mbps, which ranks as the world's fastest, whilst speeds in the United States and Sweden average 26 Mbps and 46 Mbps respectively. On the other end of the scale Venezuela has an average broadband speed of only 1 Mbps and Yemen has the slowest average speed of < 1 Mbps [27].

These bandwidth limitations in particular countries are an important point to consider based on the motivations for certain peers to acquire cryptocurrencies. For example, peers in Venezuela may wish to invest in cryptocurrencies due to current political uncertainty causing hyperinflation to the countries' fiat currency [102]. The security implications of cryptocurrency censorship will be discussed further in Section 3.7.1.

Table 3.1 summarises the different peer-to-peer bootstrapping methods and the advantages and disadvantages of each.

3.2.5 Cryptocurrency Exchanges

A brief discussion of cryptocurrency exchanges provides context regarding the pervasiveness of centralised infrastructure inherent in cryptocurrency acquisition and frames how peer-to-peer cryptocurrency bootstrapping fits into the process.

Cryptocurrency exchanges are a heavily utilised centralised infrastructure used to acquire and trade cryptocurrencies. This is in direct contradiction to the original peer-to-peer concepts outlined in the original Bitcoin paper [134]. The incongruity between the ethos of these institutions and the initial principles of Bitcoin has been the subject of other research [26, 160, 131].

The centralised nature of exchanges makes them natural targets for theft [160]. There is also strong evidence of cryptocurrency market manipulation based on the use of exchanges [92]. Despite these shortcomings, an estimated 99% of cryptocurrency trading volume is executed through exchanges [87].

Peers can avoid the use of these exchanges by participating directly on the peer-to-peer networks through the use of *core reference client software* [19]. However, we note that, whilst cryptocurrency exchanges typically execute trades off-chain because of the latency associated with clearing transactions on the blockchain, they must also eventually broadcast their transactions on-chain [77]. In order for on-chain transactions to be successfully appended to the blockchain, exchanges must also bootstrap and join as peers onto the peer-to-peer networks using core reference client software. Cryptocurrency exchanges have thus become a centralised proxy mechanism for peer connectivity.

3.3 Research Methodology Overview

In this section we present the selection process of the cryptocurrencies that we survey and outline our research steps.

3.3.1 Selection of Cryptocurrencies

Despite the volatility in the cryptocurrency market caused by new forks of well-known currencies, and new cryptocurrencies being introduced [6, 7], we needed to lock in a dataset for our research. The top 100 cryptocurrencies based on USD market capitalization was selected on February 28, 2018 [10]. These cryptocurrencies can be classified into four categories: mineable coins, non-mineable coins, mineable tokens, and non-mineable tokens, noted in Table 3.5 in Appendix 3.9.2.

The difference between coins and tokens is that coins have their own native blockchains, and tokens are built using a template on an existing blockchain [168]. For tokens, the core elements of peer-to-peer connectivity, including bootstrapping, depend on the method used by the underlying coin. Therefore, studying the bootstrapping behaviour of tokens is captured by researching the underlying coin.

Mineable and non-mineable cryptocurrencies are differentiated by their methods of acquisition. Non-mineable cryptocurrencies are generally acquired through centralised exchanges. Conversely, prior to the creation of exchanges, mineable cryptocurrencies were acquired either through direct peer-to-peer transfer or through the process of mining. The process of mining is described in the original Bitcoin paper [134].

Figure 3.1 shows the selected cryptocurrencies classified into four categories, 25 of these are non-mineable coins, 25 are mineable coins, and 50 are tokens. Non-mineable coins were excluded from our study as they depend on centralised exchanges for acquisition. That is, in relation to this study, they do not have a bootstrapping mechanism associated with peer-to-peer networking.

The 50 tokens are represented by the green hues, namely Omni, Neo, Ethereum, and Transitioning tokens. The underlying coin for the Omni tokens is Bitcoin, whilst the underlying coins for the Ethereum and Transitioning tokens is the Ethereum coin.

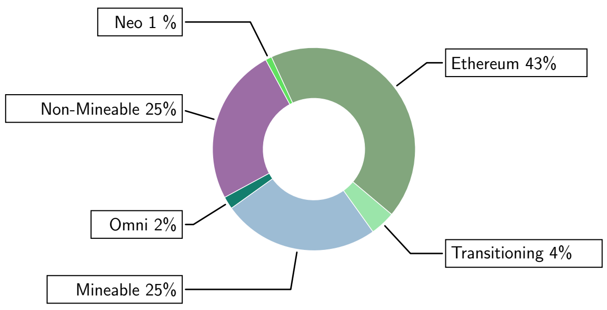

Figure 3.1: Breakdown of the Top 100 Cryptocurrencies

Ethereum tokens are also commonly referred to as ERC20 tokens. The four 'Transitioning' tokens were ERC20 tokens in the midst of migrating to their own native blockchains during our research period. Finally, the Neo token resides on an underlying coin called Gas. Gas is a non-mineable coin. Therefore, the Neo token and the Gas coin were both excluded from our research for the reasons noted previously.

In summary, our research consisted of measuring the bootstrapping methods of the 25 mineable coins, thus including the 43 ERC20, four Transitioning and two Omni tokens in our study.

3.3.2 Summary of Research Steps

A first element of our research was to survey the bootstrapping methods of 25 mineable coins and their 49 underlying tokens, noted in Section 3.3.1. The survey was completed by installing the core reference client of each mineable coin onto a virtual machine and recording its connectivity behaviour during the bootstrapping process. We then cross-referenced the behaviour against the open-source code repositories of each coin. We then iteratively tested basic censorship mechanisms, such as IP deny-lists and DNS sinkholing to determine fallback methods of bootstrapping.

Upon source code inspection in the public code repositories such as `github.com`, we uncovered that several coins have configurable means to connect via Tor. Therefore, we tested and recorded the efficacy of this bootstrapping technique. Furthermore, we tested a censorship-resistant method of bootstrapping by using ZMap to scan for potential peers and record the results.

We then performed a global measurement study over 46 geographically distributed RIPE Atlas probes in 42 countries to query the 92 DNS seeds discovered in our bootstrapping survey. RIPE Atlas is a geographically spread, volunteer-based network of

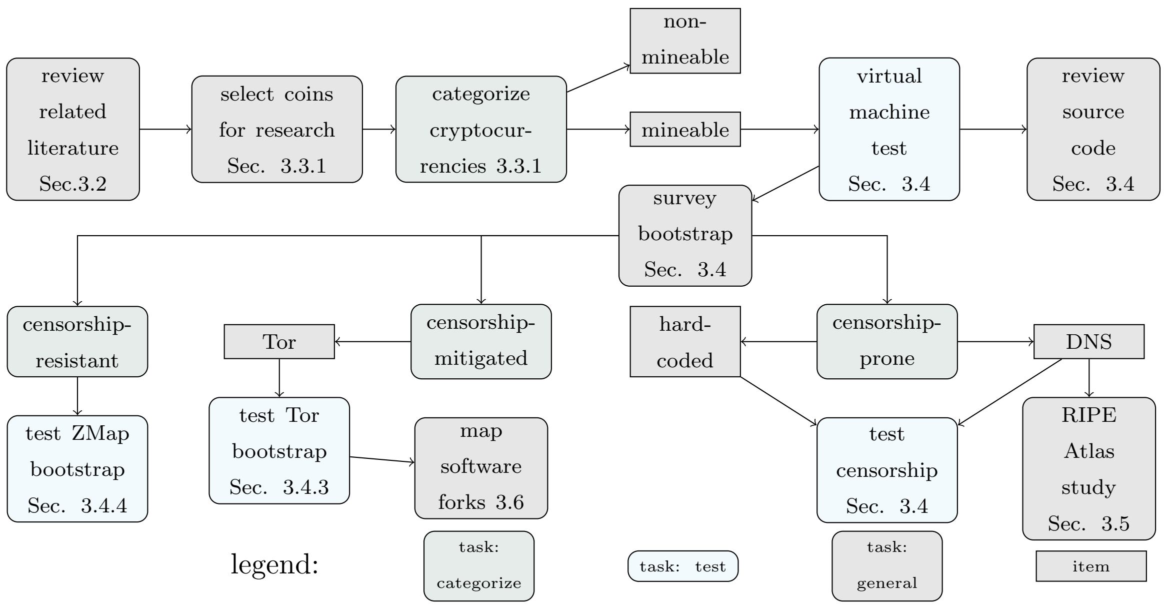
review related literature Sec.3.2
select coins for research Sec. 3.3.1
categorize cryptocurrencies 3.3.1
non-mineable
mineable
virtual machine test Sec. 3.4
review source code Sec. 3.4
survey bootstrap Sec. 3.4
censorship-resistant
test ZMap bootstrap Sec. 3.4.4
Tor
censorship-mitigated
test Tor bootstrap Sec. 3.4.3
map software forks 3.6
hard-coded
censorship-prone
DNS
test censorship Sec. 3.4
RIPE Atlas study Sec. 3.5
legend:
task: categorize
task: test
task: general
item

Figure 3.2: Research Methodology Flow Chart

probes used to measure Internet metrics such as network latency, traceroute paths, and DNS, SSL/TLS, HTTP and NTP responses [12].

In Figure 3.3 we provide a sample of what a RIPE Atlas probe may output when doing a DNS query for the domain `github.com`. It queries the Google DNS Sever with IP address 8.8.8.8, and under the ‘Answer Section’ it returns the IP address. The benefit of having a local view using globally distributed RIPE Atlas probes is that we can see certain fields, such as the IP address as seen from different locations.

Finally, as our research required the source code survey of numerous cryptocurrencies, we uncovered the root cause of inheritance of legacy bootstrapping methods by mapping the software fork lineage of the cryptocurrencies studied. In Figure 3.2 we outline the flow of our research steps and note the corresponding sections.

3.4 Survey of Cryptocurrency Bootstrapping Methods

This section presents the survey of bootstrapping methods used on the 25 mineable coins and the 49 underlying tokens selected in Section 3.3.1. The highlight of the survey indicates that 95% of the cryptocurrencies tested use only censorship-prone bootstrapping methods. We also tested and recorded the challenges of channeling peer-to-peer bootstrapping traffic via Tor and tested the feasibility of bootstrapping using ZMap IPv4

```
; <<>> DiG 9.16.1-Ubuntu <<>> @8.8.8.8 github.com
; (1 server found)
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 18148
;; flags: qr rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 0, ADDITIONAL: 1

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags:; udp: 512
;; QUESTION SECTION:
;github.com.         IN   A

;;ANSWER SECTION:
github.com.     60   IN   A   140.82.121.3

;; Query time: 23 msec
;; SERVER: 8.8.8.8#53(8.8.8.8)
;; WHEN: Mon Aug 21 15:50:00 BST 2023
;; MSG SIZE  rcvd: 55
```

Figure 3.3: Sample RIPE Atlas DNS query.

scanning.

3.4.1 Determining Bootstrapping Methods

The first task was to install the core-reference client software on 25 virtual machines and determine the bootstrapping behaviour through the use of network analysis tools and application debug logs. Next, we cross-referenced the identified behaviour with the open-source code of each cryptocurrency. As we expected, in all cases the bootstrapping connection behaviour reflects what was in the source code. We determined that three main options of peer-to-peer bootstrapping were DNS seeding, hard-coding IP seeds, and bootstrapping via Tor.

Table 3.2 records each coins bootstrapping method. Also, in Appendix 3.9.2, the tooling used to discover the bootstrapping methods is recorded in Table 3.4.

DNS seeding was the most common bootstrapping method identified; therefore, we also reviewed the DNS provider diversity for the 92 DNS seeds discovered. It is recommended that DNS name services are sourced from redundant providers to mitigate denial of service risks [33]. However, nearly one third of the coins, namely Monero, Dogecoin, Electroneum, Zclassic, Syscoin, Cryptonex, Monacoin, and Zcoin used only a single DNS provider.

Our survey allowed us to determine the characteristics of cryptocurrency peer-to-peer traffic. In Section 3.2.3 we note that countries wishing to regulate and censor Internet access attempt to block Tor traffic. Recall that Tor traffic can be identified based on destination TCP port usage, and IP connectivity to publicly published relays. Unfortunately, the same characteristics impact the ability to regulate and censor peer-to-peer cryptocurrency traffic.

95% of the cryptocurrencies evaluated in this survey use distinct TCP and UDP destination ports, making the traffic easy to identify. The ports on which cryptocurrencies operate are easily learned directly from reviewing the open-source code or from using standard network packet analysis tools. Methods to mitigate the trivial identification of cryptocurrency peer-to-peer traffic are discussed in Section 3.8.2.

3.4.2 Identifying Bootstrapping Resilience Through Basic Censorship Testing

We then proceeded to test the resilience of each cryptocurrencies bootstrapping ability by iteratively applying different censorship methods. To emulate DNS censorship, we populated the virtual machines local hosts file with the static mapping of all domain names and point them to the loopback IP address. The latter method emulated a technique known as DNS sinkholing [52]. Censorship of hard-coded seeds is achieved by IP blocking using a software firewall.

Our tests revealed that some cryptocurrencies have fallback methods of bootstrapping when the primary method is censored. We identified the fallback bootstrapping methods by first implementing DNS sinkholing or IP blocking and subsequently reviewing the application connection logs, and network packet traffic analyser outputs. Finally, we confirmed the fallback bootstrapping method by cross-referencing the relevant source code.

Seven coins were unable to bootstrap based on DNS censorship alone. These coins were Bitcoin Gold, Dogecoin, Zclassic, Syscoin, Cryptonex, Zcoin, and Vertcoin. The first four listed coins were unable to bootstrap because no secondary option of bootstrapping existed – such as hard-coded seeds. For Zclassic, there was a configuration option for a fallback bootstrap mechanism via hard-coded seeds, but no values existed. In the case of Zcoin, and Vertcoin, connection attempts were seen to hard-coded IPs which were verified in the source code. However, no successful connections were completed after a period of 24 hours.

Five coins were unable to bootstrap based on IP censorship alone. These coins were Ethereum, which also covers 47 underlying tokens, Ethereum Classic, Siacoin, and Bitcore. Therefore, if we consider the underlying tokens, 80% of the cryptocurrencies tested did not have resilient bootstrapping methods configured. Our tests revealed that if DNS censorship is combined with IP censorship, that only Verge was able to bootstrap by default through the use of Tor.

Furthermore, Syscoin was unable to reconnect to the peer-to-peer network when DNS seed censorship was present. DNS censorship should have limited success for peer reconnection because in typical operation previously connected peers should be locally cached [95]. This indicated a persistent dependency on DNS for ongoing connectivity requirements, rather than limiting the dependency on this censorship-prone resource for bootstrapping alone.

Table 3.2 summarises the results of this section and the fallback methods of bootstrapping that we discovered.

3.4.3 Results of Tor Bootstrapping

Reviewing the source code for 25 cryptocurrencies uncovered the option to support connectivity through a proxy server for 20 of the coins. Therefore, we tested the ability to bootstrap via Tor. This allowed us to explore the rate of success when bootstrapping through a censorship-mitigated method. Verge was not included in testing because it uses Tor by default.

We performed our test by installing the Tor expert bundle or torsocks on the Windows or Ubuntu virtual machines and configure the local SOCKS proxy to channel the cryptocurrency peer-to-peer traffic. We also ensured the local peer caches were cleared to ensure a true bootstrapping experience.

Only 13 coins were able to successfully bootstrap, five were unsuccessful, and two bootstrapped but had various issues. The five coins unable to bootstrap were Bitcoin Gold, Zcash, Bytecoin, Dogecoin, and Electroneum. The two coins which bootstrapped but had issues are Bitcoin Cash and Monero. For Bitcoin Cash finding other peers to connect to for bootstrapping was successful. However, after 24 hours of operation the estimated time to download the full blockchain through Tor was quoted as one year and 27 weeks, making this method of connectivity infeasible in practice. Monero was able to proxy some of its peer-to-peer traffic via Tor, but the UDP based DNS queries leak outside of the SOCKS proxy and setting the DNS_PUBLIC=tcp option resulted in several errors with connectivity. The README.md on the 'Using Tor' section of Monero, Electroneum, and Bytecoin explicitly indicated that these coins were not meant to integrate with Tor [8, 5]. Therefore, any issues experienced were within the expectations set by their core developers. We discuss the reasons for this similarity in Section 3.6.

Although bootstrapping via Tor can mitigate the risk of censorship, it may not provide the levels of anonymity desired [41, 90]. Also, proxied connections must accept latency and performance penalties limiting the ability to download large blockchains in a timely manner. Furthermore, in a highly regulated environment, extra caution is required to avoid the censorship of Tor itself. This includes the use of pluggable transports to obscure the character of Tor traffic and the manual collection of private bridge IP addresses. The results in this section are summarised in Table 3.2.

3.4.4 Results of ZMap Bootstrapping

In the previous sections we have surveyed the censorship-prone methods of bootstrapping present in the majority of cryptocurrencies researched, and we also tested their resilience to basic censorship to determine fallback methods of bootstrapping. We also tested the ability to bootstrap through a censorship-mitigated method via Tor. In this section we present the results when we tested the feasibility of bootstrapping through a censorship-resistant method by using scanning the IPv4 address space using ZMap. The majority of core reference software peers limit their connections to eight other peers. Therefore, on our ZMap scanning parameter we configured the scan to halt as soon as eight peers with the relevant destination TCP ports were discovered. We first recorded the amount of time it took to discover the peers using ZMap, then we tested the ability to bootstrap using the discovered peer IPs.

Our results showed that the mean time to discover eight peers was 16min 29sec. Surprisingly, none of the cryptocurrencies could bootstrap successfully using this method. As this result was unexpected, we attempted to run a full IPv4 scan on port tcp/8333 (the port used by Bitcoin). The scan took just under four hours to complete and only discovered 241 IPs, none of which allowed Bitcoin to successfully bootstrap. Also, Siacoin has a list of 101 hard-coded seeds configured in the source code, therefore we ran an exhaustive scan to see if any of these seeds were discovered. Regrettably, although 221 IPs were found listening on Siacoins' destination TCP port (with a total scan time

Table 3.2: Cryptocurrency Bootstrap Methods and Connection Results. Tor Option = Tor connectivity configurable, Tor Outcome = details Tor bootstrap, DNS Censor = DNS censorship able to prevent bootstrap, DNS Single = DNS seeds have multiple providers, Config Issues = other issues, ZMap Time = time in mm:ss to discover 8 peers, ZMap Bootstrap = bootstrap success, ● = yes, ○ = no, × = N/A, ⊙ = successful bootstrap, ◎ = bootstrap has issues, ⊗ = bootstrap not successful after 24h, ⋄ = GUI option for Tor, †= DNS seed configured, yet NXDOMAIN returned, ‡= hard-coding misconfiguration, (#) = number of underlying tokens.

Cryptocurrency	**Ticker**	**Bootstrap**	**Tor Option**	**Tor Outcome**	**DNS Censor**	**DNS Single**	**Config Issues**	**ZMap Time**	**ZMap Bootstrap**
Bitcoin (2)	BTC	DNS Seed Hard-Coded	●	⊙	○	○	○	7:58	○
Ethereum (47)	ETH	Hard-Coded	○	×	×	×	○	×	×
Bitcoin Cash	BCH	DNS Seed Hard-Coded	●	◎	○	○	○	7:58	○
Litecoin	LTC	DNS Seed Hard-Coded	●	⊙	○	○	○	23:43	○
Dash	DASH	DNS Seed Hard-Coded	●	⊙	○	○	○	26:32	○
Monero	XMR	DNS Seed Hard-Coded	●	◎	○	●	†	18:43	○
Ethereum Classic	ETC	Hard-Coded	○	×	×	×	○	15:54	○
Bitcoin Gold	BTG	DNS Seed	●	⊗	●	○	○	16:08	○
Zcash	ZEC	DNS Seed	●	⊗	○	○	○	32:15	○
Bytecoin	BCN	Hard-Coded	●	⊗	×	×	○	29:24	○
Verge	XVG	Tor	●	⊙	×	×	○	×	×
Dogecoin	DOGE	DNS Seed	●	⊗	●	●	○	29:29	○
Siacoin	SC	Hard-Coded	○	×	×	×	○	11:42	○
Electroneum	ETN	DNS Seed Hard-Coded	●	⊗	○	●	†	10:59	○
HShare	HSR	DNS Seed Hard-Coded	●	⊙	○	○	†	18:27	○
Zclassic	ZCL	DNS Seed Hard-Coded	●	⊙	●	●	‡	16:41	○
Komodo	KMD	DNS Seed Hard-Coded	●	⊙	○	○	○	6:50	○
Syscoin	SYS	DNS Seed	●	⊙	●	●	○	6:45	○
Digibyte	DGB	DNS Seed Hard-Coded	●	⊙	○	○	○	15:45	○
Cryptonex	CNX	DNS Seed	○	×	●	●	○	7:16	○
Monacoin	MONA	DNS Seed Hard-Coded	●	⊙	○	●	○	14:58	○
Bitcore	BTX	Hard-Coded	●	⊙	×	×	○	11:20	○
Zcoin	XZC	DNS Seed Hard-Coded	●⋄	⊙	●	●	○	28:46	○
Vertcoin	VTC	DNS Seed Hard-Coded	●	⊙	●	○	○	7:58	○
SmartCash	SMRT	DNS Seed Hard-Coded	●	⊙	○	○	○	13:32	○

exceeding five hours), none of these IPs coincided with the hard-coded seeds. Consequently, bootstrapping was not successful using any of the discovered IPs. Because none of the cryptocurrencies could bootstrap using eight discovered IPs, we also ran a series of exhaustive scans and noted scan times would typically take between four to five hours to complete, making this method infeasible in practise. The scan times would also be impacted by the variance in geographical network latency, noted in Table 3.3.

We could surmise that one of the challenges impacting this method of bootstrapping was the bandwidth requirements to run a full scan in a timely manner [2]. Furthermore, we noted an issue could arise when destination port numbers were commonly used by other services. For example, Bytecoin uses port tcp/8080, a commonly used port for proxy and web services. Furthermore, Vertcoin, Bitcoin Cash, and Bitcoin all use the same destination port, so finding peers unique to the particular peer-to-peer network would be fraught with issues. Finally, Ethereum uses dynamic ports for peer connectivity, therefore, this method of peer discovery was not compatible with this coin.

Despite the major advantage that this method of bootstrapping requires no centralised resources, our results show that this is not a feasible method of bootstrapping due to the nil success rate, varying discovery times, and the fact that the peer-to-peer traffic is still identifiable by the destination port of its operation. Results for this section can be seen on Table 3.2.

3.5 Global RIPE Atlas Study on DNS Seeds

After determining the bootstrapping methods of 25 coins and 49 underlying tokens, we now turn our investigation to the measurements obtained from our global RIPE Atlas DNS seed bootstrapping study. We first provide statistics and insights about the distribution of IP addresses returned from the DNS query responses. Then we expose several negative results impacting DNS seed bootstrapping, such as outages and query response manipulation.

The research presented in this section stems from an objective to systematically assess the DNS-based method of bootstrapping modern peer-to-peer networks. By rigorously examining DNS query response distributions and unveiling instances of disruptions and

[2]We also came across another practical issue during the testing of Zmap. We initially tested Zmap working in a co-working office environment. Within moments of initialising this powerful scanning tool, all the bandwidth in this office was completely saturated. This shortcoming could also impact the efficacy of Zmap as it exhausts bandwidth at a rapid rate rendering all other work on the same connection infeasible.

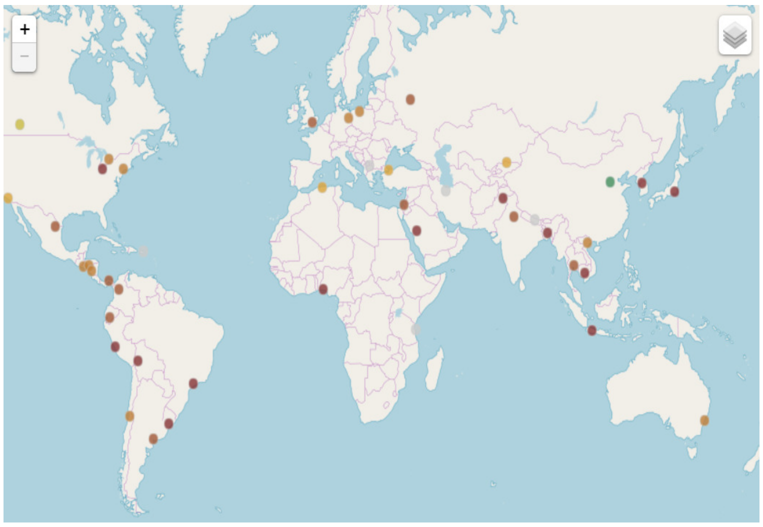

Figure 3.4: RIPE Atlas Probe Locations. The colours represent the latency associated with each DNS query response when this RIPE Atlas image was collected in our 2018 measurement study: red = high, amber = medium, green = low, grey = no data. The latency times are also summarised in Table 3.3 under the column labelled Latency.

response tampering, our goal is to contribute robust insights essential for bolstering the reliability and integrity of peer-to-peer infrastructures.

3.5.1 Detailed Research Methodology of Study

We begin by elaborating on the research steps outlined in Section 3.3 which are specific to this element of our study.

RIPE Atlas Probe Selection

To begin our RIPE Atlas study we needed to select the RIPE Atlas probes that would be used to perform the DNS queries for our research. Based on a data collection interval of two months we choose 46 RIPE Atlas probes from 42 countries across Asia Pacific, Europe Middle East Africa, North America, and South America.

In several countries trading and acquiring cryptocurrencies is illegal or is under legal scrutiny [154, 9]. Therefore, where possible, we selected RIPE Atlas probes from these countries with the intent of measuring possible DNS response manipulation. A summary of the countries selected and their legal standpoints towards cryptocurrencies can be seen in Table 3.3. A longitudinal view of each RIPE Atlas probe in our study can be seen in Figure 3.4.

DNS Seed Enumeration and Selection

Our next step was to enumerate all the DNS seeds used by the cryptocurrencies. These are noted in the source code and were verified on the virtual machines, outlined in Section 3.4. Only responsive seeds were selected for analysis.

DNS query responses include Return Codes which indicate the status of the response. The Return Codes seen in the RIPE Atlas study are 0 – NOERROR, indicating that the query is successfully completed, 2 – SERVFAIL indicating that a server failure has occurred to the DNS query, and 3 – NXDOMAIN indicating that the requested domain name does not exist.

Return Code manipulation is a typical DNS censorship technique [25, 140]. For example, if cryptotest1.io was configured with an A record, the correct query response would include Return Code 0, indicating NOERROR, along with the correct IP address. However, in a censored environment, the query response may be manipulated to Return Code 3, indicating NXDOMAIN (i.e., saying the requested domain does not exist). This would result in withholding the IP address from the client and ultimately preventing access to the resource. In Section 3.5.3 we outline an example of Return Code manipulation witnessed in our study.

Data Collection Intervals

For the majority of DNS seeds, the RIPE Atlas data collection was conducted between May 6 – July 6, 2018. From May 6 – June 2 each seed was queried on each probe every 24 hours. We were able to increase the interval to every six hours from June 3 – July 6 because we accrued more RIPE Atlas credits to support an increased polling interval for our measurement study.

Due to initial struggles with the verification of the DNS seed bootstrapping process for the coins HShare, Komodo, Zcash and Zclassic, the DNS seeds for these coins were tested between August 12 – September 23. The polling interval for these coins was every six hours. Instead of excluding these coins from our study due to issues with source code verification, we opted to perform the same manner of RIPE Atlas data collection on these coins, albeit during a different time period.

Data Processing Steps

The raw data from the DNS queries made from the RIPE Atlas probes was output into a JavaScript Object Notation (JSON) format. We extracted the fields relevant to our study using a Python script and output this data into a CSV which was then further processed into an SQL script with the appropriate INSERT statements to input into a MySQL relational database. The data queried from the MySQL database forms the basis of our measurement study. All the tooling used for our research can be found in the chapter Appendix on Table 3.4.

3.5.2 DNS Seed Measurement Results

In this section we present the main results of our global measurement study into DNS seeds used for cryptocurrency bootstrapping. We will capture the statistics surveyed for each cryptocurrency, highlight peers active across multiple coins, and reveal the variance in distinct IPs recorded in countries with differing legal standpoints towards these digital assets.

IP Statistics Per Cryptocurrency

We begin our data analysis by capturing the statistics regarding the number of IP addresses returned for each cryptocurrency. In Figure 3.5 we record the number of distinct IPs and total number of IPs of the cryptocurrencies that we researched. To understand the difference between the set of distinct and total IPs for each cryptocurrency we provide the following example.

Example 1. This example uses private IPs, therefore no information about real peers is disclosed.

Assume two DNS queries are made for the DNS seed exampleseed.mycryptocurrency.org. The first query response returns the set of IPs $\mathcal{S}_1 = \{10.0.0.1, 10.0.0.2, 10.0.0.3, 10.0.0.4\}$, and the second query response returns the set of IPs $\mathcal{S}_2 = \{10.0.0.1, 10.0.0.5\}$. The number of total IPs from the two queries is simply $|\mathcal{S}_1| + |\mathcal{S}_2| = 4 + 2 = 6$. The number of distinct IPs from the two queries is $|\mathcal{S}_1 \cup \mathcal{S}_2| = |\{10.0.0.1, 10.0.0.2, 10.0.0.3, 10.0.0.4, 10.0.0.5\}| = 5$.

Our measurement data reveals that the following coins returned less than 26 distinct IPs: Monacoin (18), Cryptonex (21), Zcoin (24), and Zclassic (25). Conversely, the remaining coins with the highest number of distinct IP addresses returned are: Bitcoin (44077), Litecoin (7393), Dash (5489), Dogecoin (2812), Bitcoin Cash (2512), Syscoin (1280), Vertcoin (967), Zcash (619), Bitcoin Gold (331), Digibyte (295), Komodo (86), and SmartCash (67).

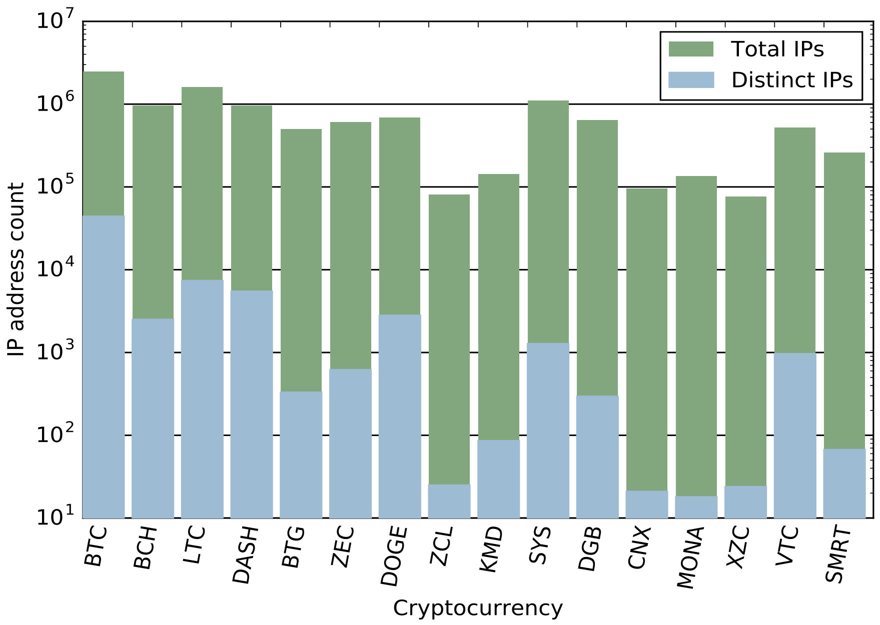

Figure 3.5: Number of Total and Distinct IPs from RIPE Atlas DNS Seed Query Responses. The y − axis is logarithmic.

For the coins returning less than 26 distinct IPs, the bootstrapping process relies on a static set of servers for connection to the peer-to-peer network. That is, the DNS query responses reveal the IPs of rendezvous servers, rather than the IPs of other peers on the network. Unfortunately this bootstrapping method reflects the legacy method of how Napster peers connected to the peer-to-peer network, noted in Section 3.2.2.

The coins returning less than 26 distinct IPs have a disadvantage in terms of being able to withstand basic censorship. However, they inadvertently prevent side-channel information leakage. In contrast, consider coins returning a large set of dynamically changing IPs in the DNS query responses. A clear side-channel is exposed to harvest information about the IPs of peers. This information can be collected without the need to install any core reference client software. That is, DNS query responses reveal the IPs of peers on the network, which is an unintended consequence of DNS seeding.

Peers Active Across Two or More Cryptocurrencies

Our measurement study reveals that 5.2% of the distinct IPs identified are active across two or more cryptocurrencies. We calculate this result as follows: Let $\mathcal{C}$ be the set of cryptocurrencies which use DNS seeding as a bootstrap method, where:

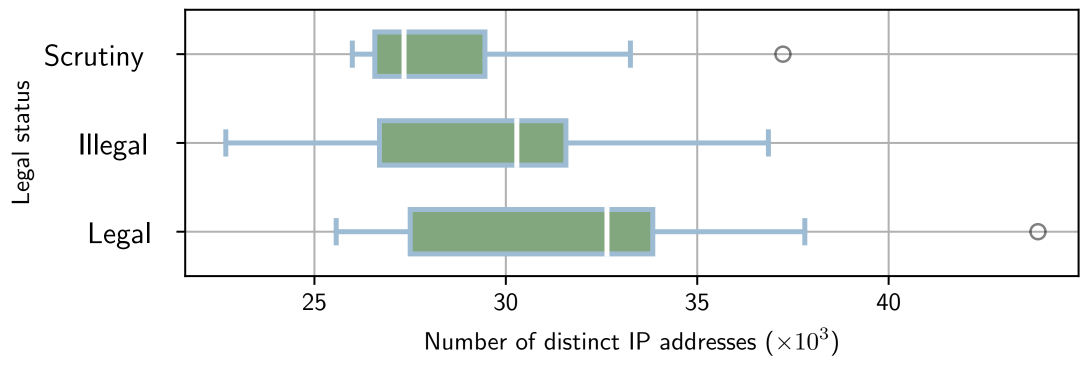

Figure 3.6: Distinct IP Address Count Statistics of Countries Grouped by Cryptocurrency Legal Status.

$\mathcal{C} = \{\mathsf{BTC}, \mathsf{BTH}, \mathsf{LTC}, \mathsf{DASH}, \mathsf{XMR}, \mathsf{BTG}, \mathsf{ZEC}, \mathsf{DOGE}, \mathsf{ETN}, \mathsf{HSR},$ $\mathsf{ZCL}, \mathsf{KMD}, \mathsf{SYS}, \mathsf{DGB}, \mathsf{CNX}, \mathsf{MONA}, \mathsf{XZC}, \mathsf{VTC}, \mathsf{SMRT}\}$. Let $\mathcal{D}_i$ be the set of distinct IPs returned for each $i \in \mathcal{C}$. Our data shows that:

$$\sum_{i \in \mathcal{C}} |\mathcal{D}_i| = |\mathcal{D}_{\mathsf{BTC}}| + |\mathcal{D}_{\mathsf{BTH}}| + |\mathcal{D}_{\mathsf{LTC}}| + \ldots + |\mathcal{D}_{\mathsf{SMRT}}| = 66016$$

If we let $\mathcal{I}$ represent the set of distinct IPs in our entire data set, we found that $|\mathcal{I}| = 62732$. This reveals that:

$$\sum_{i \in \mathcal{C}} |\mathcal{D}_i| - |\mathcal{I}| = 3284$$

That is, 3284 out of the 62732 peers measured are active across two or more cryptocurrencies. IPs active across multiple cryptocurrencies could be individual peers with diversified investments. However, they could be the IPs of cryptocurrency exchanges, which act as a proxy for numerous peers. Identifying and censoring the IPs used by exchanges would have a profound impact to the trade and acquisition of these digital assets for numerous peers.

IP Statistics by Legal Status

The final statistic we capture is the number of distinct IP addresses seen in countries grouped by their legal status towards cryptocurrencies, noted in Table 3.3. In Figure 3.6 we see that lower distinct IP counts tend to be in countries where cryptocurrencies are illegal or are under legal scrutiny, and that the higher IP counts belong to countries where cryptocurrency ownership is fully legal.

3.5.3 DNS Manipulation

In this section we present results regarding evidence of manipulation of the cryptocurrency bootstrapping process based on the reliance on DNS seeding. To maintain privacy, we only disclose the AS (Autonomous System) number containing the peer IPs identified in our study.

NXDOMAIN Manipulation

In our study we found that three probes injected manipulated results for DNS query responses on Non-Existent Domains. All other probes return NXDOMAIN, but these probes return NOERROR. The ability to manipulate the return code in a DNS query response and present fabricated peer IPs reflects the ability to modify the bootstrapping process. Presenting false peer IPs could isolate the peer from the peer-to-peer network. Worse, manipulated IPs could resolve to a reconnaissance host which records connection attempts to the cryptocurrency peer-to-peer network [119]. This is concerning for peers in countries where cryptocurrencies are illegal, as noted in Table 3.3.

The behaviour of manipulating DNS query responses in China is well documented [25, 140]. However, we surprisingly record this behaviour in Brazil and in the USA. All the DNS seeds for Monero return NXDOMAIN on all probes, except for China – which is expected. Interestingly, we also see return code manipulation on a probe in Yorkton Heights, USA. The latter probe returns an IP in AS45028 for seeds.moneroseeds.ae.org. The IP is reachable, but it suspiciously does not accept connections to TCP ports 18080 or 28080 on which Monero operates. The China probe returns an IP in AS4837, which is unreachable. It also returns the same IP when other probes report NXDOMAIN for DNS seeds: dnsseed.dashpay.io, node-singapore.cryptonex.org, and seeds.electroneum.com. The Yorkton Heights probe also exhibits the same manipulated NXDOMAIN behaviour on seed1.smartcash.org, seed2.smartcash.org, node-singapore.cryptonex.org, and the domain seeds.electroneum.com. Finally, on the seed singapore.cryptonex.org we see the Brazil host also returning a false NOERROR. Results are summarised in Table 3.3.

TTL Manipulation

In this section we identify a commonly manipulated parameter in DNS query responses called the Time to Live (TTL) value. We show examples of this manipulation captured in our study and analyse how this manipulation impacts cryptocurrency bootstrapping behaviour.

TTL values tell DNS servers how long to cache query responses in order to reduce network traffic and improve performance. The prevalence of TTL manipulation globally has been noted to occur in 20% of DNS query responses [140]. The main intention behind TTL manipulation is generally motivated by performance considerations.

Example 2. This example uses a private IP, ensuring no information about real resources is revealed.

Assume a website ttlexample.earth has a single A record referencing IP address 10.0.0.50 with a TTL of 3600s. To reduce DNS query traffic an ISP may manipulate the TTL from 3600s to 86400s. By manipulating the TTL, the IP address for this website is cached for 23 hours longer than the authentic parameter. As a result DNS query traffic is minimised for this record over a 24-hour period.

For websites returning a static set of IP addresses TTL manipulation decreases traffic load and increases performance. However, although well intended, TTL manipulation may not be suitable for domains that host dynamic IP content, such as the coins we see in Section 3.5.2, which return peer IP addresses. Considering the churn rate associated with peer-to-peer networks [117], manipulating TTL values to high values could also lead to the return of stale peer IPs which are no longer active on the network, thereby adversely impacting the bootstrapping process.

During the period of our research, the data reveals that TTL manipulation would have caused peer bootstrapping behaviour changes for: Dogecoin, Bitcoin, Vertcoin, and Bitcoin Gold. For Dogecoin the primary DNS seed seed.multidoge.org returns SERVFAIL thereby returning no peer IP addresses on 93% of our selected probes between June 2 – 16. Yet on probes in El Salvador, Honduras and Ohio, USA, we still see NOERROR responses between June 2 – 4 because of overwritten TTL values. Luckily, the Dogecoin secondary seed seed2.multidoge.org remains responsive during this period otherwise bootstrapping would have failed because Dogecoin does not have a diverse bootstrapping mechanism configured.

For Bitcoin, the DNS seed dnsseed.bitcoin.dashjr.org returns a SERVFAIL for all probes during the May 11 – June 2, 2018 period. However due to TTL manipulation, the probes in China, El Salvador, Honduras, and Panama continue to return results between May 11 – 12, and Australia continues to return results between May 11 – 24. For Bitcoin, outage on this seed dnsseed.bitcoin.dashjr.org is not a major issue impacting bootstrapping behaviour because all other seeds were returning results during this period. Also, Bitcoin provides a large list of hard-coded IP addresses for bootstrapping as a fallback method, as noted on Table 3.2.

Table 3.3: Details of RIPE Atlas Probes. This table records the country and region of each probe, the associated legal status towards cryptocurrencies, the number of probes chosen in each country, and the latency of the DNS query responses. The table also highlights if the probe(s) in that country experienced issues related to common DNS manipulation techniques and other DNS related issues. Abbreviations: NXD = NXDOMAIN manipulation present, TTL = TTL manipulation present, Google = Google DNS issues present, ● = yes, ○ = no, † = average, ‡= probe went offline June 28, 2018.

Country	**Region**	**Status**	**# of probes**	**Latency < x ms**	**NXD**	**TTL**	**Google**
Algeria	EMEA	illegal	1	50	○	●	○
Argentina	SAMER	legal	1	30	○	●	○
Australia	APAC	legal	1	20	○	●	○
Bangladesh	APAC	illegal	1	100	○	●	○
Bolivia	SAMER	illegal	1	100	○	○	○
Brazil	SAMER	legal	1	200	●	○	○
Cambodia	APAC	scrutiny	1	800	○	○	●
Canada	NAMER	legal	2	60†	○	○	●
Chile	SAMER	legal	1	40	○	○	○
China	APAC	scrutiny	1	10	●	●	○
Columbia	SAMER	legal	1	200	○	●	○
Ecuador	SAMER	illegal	1	200	○	○	●
El Salvador	SAMER	legal	1	50	○	●	○
Germany	EMEA	legal	1	40	○	○	●
Honduras	SAMER	legal	1	100	○	●	○
India	APAC	scrutiny	1	100	○	○	○
Indonesia	APAC	scrutiny	1	30	○	○	○
Iran	EMEA	illegal	1	no data	○	○	○
Israel	EMEA	legal	1	200	○	○	●
Japan	APAC	legal	1	10	○	○	○
Kyrgyzstan	APAC	legal	1	200	○	○	●
Macedonia	EMEA	illegal	1	no data	○	●	○
Mexico	NAMER	legal	1	100	○	○	○

Continued on next page

Continued from previous page							
Country	**Region**	**Status**	**# of probes**	**Latency < x ms**	**NXD**	**TTL**	**Goog.**
Nepal ‡	APAC	illegal	1	no data	○	○	○
New Zealand	APAC	legal	1	20	○	○	○
Nicaragua	SAMER	legal	1	100	○	○	●
Nigeria	EMEA	scrutiny	1	100	○	○	○
Pakistan	APAC	scrutiny	1	200	○	○	●
Panama	SAMER	legal	1	200	○	●	●
Peru	SAMER	legal	1	100	○	○	●
Poland	EMEA	legal	1	20	○	○	○
Puerto Rico	SAMER	legal	1	no data	○	○	○
Russia	EMEA	scrutiny	1	40	○	○	○
Saudi Arabia	EMEA	legal	1	100	○	○	○
South Korea	APAC	legal	1	200	○	●	○
Tanzania	EMEA	legal	1	10	○	○	●
Thailand	APAC	scrutiny	1	40	○	○	○
Turkey	EMEA	scrutiny	1	200	○	○	●
United Kingdom	EMEA	legal	1	30	○	○	○
Uruguay	SAMER	legal	1	30	○	●	○
USA	NAMER	legal	4	50[†]	●	●	●
Vietnam	APAC	scrutiny	1	100	○	○	●

We also see the seed dnsseed.pknight.ca for Vertcoin unresponsive between July 4 – 6. Yet, due to TTL manipulation we continue to see DNS responses returning peer IP addresses in China, El Salvador, Honduras and Panama. At this time Vertcoin was also in a precarious position. It only had one other seed responsive during this period and, as noted in Section 3.4, the hard-coded IP addresses in Vertcoin's secondary bootstrapping configuration are unreachable. Finally, an outage in Bitcoin Gold DNS seeds, which we cover in more detail in Section 3.5.4, highlights TTL manipulation in Algeria, Argentina, Bangladesh, China, Columbia, El Salvador, Honduras, Macedonia, South Korea, Uruguay and the USA. In all of these cases, assuming the peer IPs were active, TTL manipulation provided the increased availability of DNS seeds that would have otherwise been unavailable from the DNS servers used by the probes in the countries noted.

However, the ability to manipulate TTL values on cryptocurrency DNS seeds could also provide negative consequences. Manipulating TTL values too high could isolate a bootstrapping peer from the network if all the returned IP addresses belonged to peers no longer active on the network. For example, measurement studies of Bitcoin nodes indicate that the majority of IP addresses are active less than five days [72]. Therefore, for Bitcoin, manipulating a TTL to a value that exceeds five days could return stale peer IPs and interrupt the bootstrapping process. Excessive TTL manipulation would have the most profound impact on the coins in Table 3.2 which do not have a fallback bootstrapping method. The results of this section are summarised in Table 3.3.

3.5.4 Recorded Outages and Issues

In this section we review the DNS seed outages impacting cryptocurrency bootstrapping during our study. We identified an outage impacting Bitcoin Gold and highlighted issues specific to Google DNS servers across 14 countries impacting a total of six cryptocurrencies.

Bitcoin Gold Outage

Our measurement study reveals a near miss outage for bootstrapping on Bitcoin Gold between June 22 – 23, 2018. Bitcoin Gold has two responsive DNS seeds: dnsseed.btcgpu.org and dnsseed.bitcoingold.org. We see that for IPv6 responses both seeds return SERVFAIL codes, and that for IPv4 responses, dnsseed.btcgpu.org also returns SERVFAIL codes for all probes between 19:00UTC June 22 and 18:00UTC June 23. During this time period a single IPv4 address in AS22612 is returned on the dnsseed.bitcoingold.org seed. Typically, each query response for this seed returns 29 peer IPs. However, during this period, bootstrapping onto the Bitcoin Gold depends on a single IP address. This situation is especially precarious for Bitcoin Gold, as Table 3.2 indicates DNS seeding is the only bootstrapping option configured for this coin.

Google DNS Issues

On June 30, 2018 we also find that Google DNS servers across 14 countries – noted on Table 3.3 – return no IPv4 addresses across the DNS seeds of six cryptocurrencies: Zcoin, Monacoin, Litecoin, Digibyte, Dash, and Cryptonex. In all cases a NOERROR response code is provided but no peer IPs are returned. The issue appears to relate to seeds in the top-level domains .io and .org. Furthermore, for Bitcoin Gold, during the entire data collection interval from May 6 – July 6, the exact issue is seen in the same 14 countries using Google DNS. IPv6 records are returned during this time (with the exception of the outage time noted in Section 3.5.4). Therefore, Bitcoin Gold peers in these regions using Google DNS with limited IPv6 functionality would struggle to bootstrap under these conditions.

This section has highlighted several negative consequences of using DNS seed bootstrapping for cryptocurrencies. By depending on DNS, we see that bootstrapping is subject to connection manipulation and outages in 60% of the countries that we investigate. The outages recorded relate to operational issues on individual coins and the use of specific DNS providers.

3.6 Uncovering Bootstrapping Inheritance

During our research we concurrently reviewed the source code of 25 cryptocurrency coins and noted the considerable similarities in the style and syntax of the code. Based on this finding we completed an analysis of the code repositories to map the software lineage (known as a software forks) of each cryptocurrency researched. This uncovered a chain of inheritance of each source code repository to only five main code bases.

The software forks identified the predominance of Bitcoin source code in over two thirds of the coins. The similarity of bootstrapping methods in the software forks of Bitcoin can be traced to the chainparams.cpp file. The source code reveals that Bitcoin elects to use DNS seeding as the primary bootstrapping method and we see that the core developers of the software forks of Bitcoin merely inherit the legacy bootstrapping mechanisms as an oversight rather than an explicit design decision.

In Figure 3.7 we summarise our findings of software fork lineage. We reflect that the software forks represent the inheritance of source code which includes suboptimal elements such as censorship-prone bootstrapping. The figure also illustrates any hard forks of the 25 cryptocurrencies that we research, and we refer the interested reader to Appendix 3.9.1 for further details. A total of five main colour hues exist on the diagram representing the original source code parents for the cryptocurrencies: blue–Bitcoin (17), green–Bytecoin (3), orange–Verge (2), brown–Ethereum Classic (2), and yellow–Siacoin (1), where the number in the brackets indicates the number of coins having the same source code lineage. With the exception of Verge, the other four source code parents use legacy bootstrapping methods. Also, the cryptocurrencies Litecoin, Monero, Zcoin, and

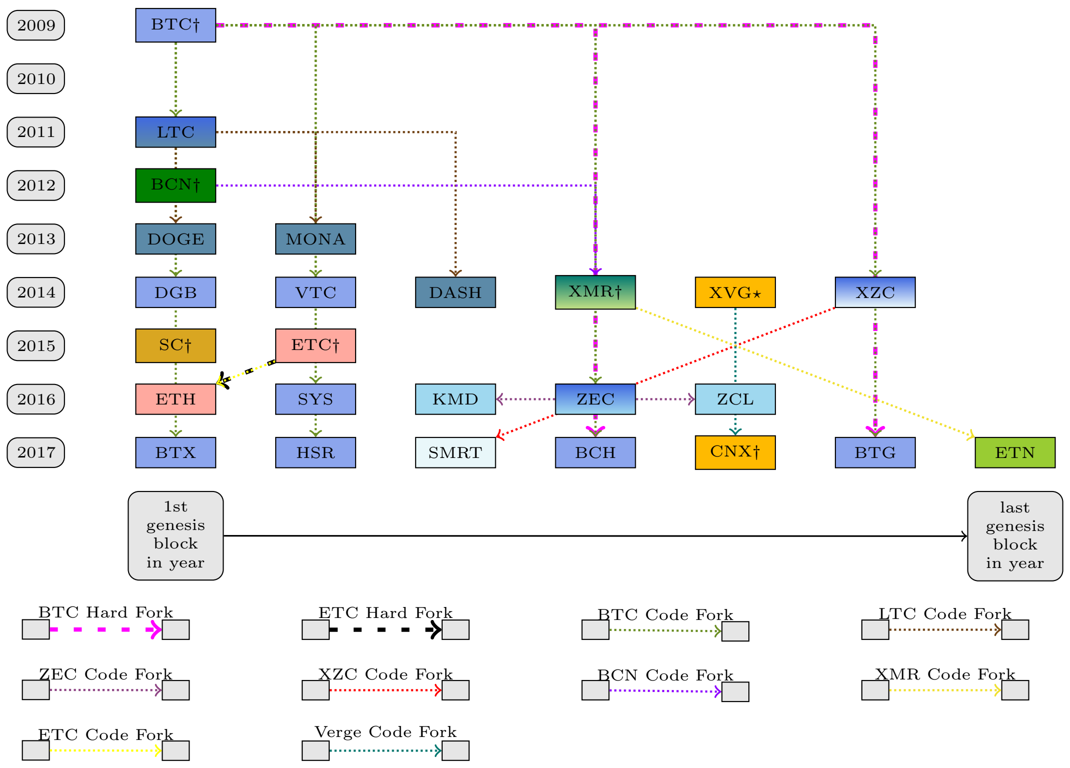

Figure 3.7: Cryptocurrency Hard Forks and Software Forks. †= configured censorship-prone bootstrapping explicitly, ⋆ = configured censorship-mitigated bootstrapping explicitly, no symbol = inherited bootstrapping method.

Zcash have two shades to indicate that they are both software fork children and software fork parents.

To the best of our knowledge, through our measurement study we provide the first insight into the lack of variance in cryptocurrency source code. As well as inheriting bootstrapping methods we note that any other vulnerability disclosure may materially impact coins beyond the individual cryptocurrency being researched.

3.7 Implications of Findings

In this section we explore the security and social implications of our findings which are specific to cryptocurrencies. Our findings are new with respect to previous measurement and censorship studies [21, 162]. In contrast to the outcomes of censoring file sharing networks, which mainly share digital music and videos, we highlight that the consequences of censoring cryptocurrencies are more profound.

3.7.1 Security Implications of Findings

Based on the results of our measurement study we summarise the security implications discussed throughout the chapter. Whilst there have been many other Internet measurement and censorship studies, our work is the first to focus on the simultaneous measurement of a large group of cryptocurrencies.

We have highlighted that 95% of cryptocurrencies surveyed use DNS seeding and/or IP hard-coding as methods to bootstrap. Therefore, basic methods of censorship, such as DNS sinkholing of IP deny-lists are effective at preventing peer connectivity to cryptocurrency networks. With regards to DNS seeding, we see that four cryptocurrencies return less than 26 distinct IPs, highlighting the use of rendezvous servers for peer connectivity. With this in mind, the high degree of peer bootstrap centralisation leaves these currencies more susceptible to outages and censorship.

Conversely, we see that ten cryptocurrencies return sets of hundreds or even thousands of peer IPs in their DNS query responses, thus revealing an unintended side-channel to harvest information about cryptocurrency peers. The ability to harvest cryptocurrency peer IPs via DNS query responses brings an interesting consideration concerning personal data protection laws, such as the European Union's General Data Protection Regulation (GDPR). GDPR applies to the processing of personal data of individuals within the EU [3]. According to the regulation, if information is provided which enables the identity of an individual behind an IP then it is considered personal data [2]. It is possible to combine the data from time stamped DNS query responses with ISP logs to potentially identify individuals, or groups of individuals using the IP recorded.

Additionally, 32% of cryptocurrencies do not use redundant DNS providers for their DNS seeds, leaving themselves exposed to a single point of DNS service failure. Furthermore, evidence of DNS manipulation is seen due to the lack of DNSSEC on any of the cryptocurrencies surveyed. This could lead to peers in countries where cryptocurrencies are illegal being identified through the use of a reconnaissance host IP injected into the DNS query response. Additionally, due to the churn associated with peer-to-peer networks, TTL manipulation could constrain peer connection due to stale records being cached at ISP DNS servers. Also, 88% of the cryptocurrencies surveyed use distinctive destination ports, leaving them open to basic port-based censorship.

[3]While GDPR specifically applies to individuals within the EU, certain aspects of the regulation, particularly those related to the protection of personal data, have broader implications. Even for non-EU citizens, adherence to GDPR principles regarding personal data protection could also be recognised as a global standard for ethical handling of user information.

Our study has also revealed that bootstrapping via Tor and ZMap scanning are not always feasible alternatives to censorship-prone options. This highlights that a defence-in-depth approach to cryptocurrency bootstrapping should be applied. Finally, we reveal the lack of variance in cryptocurrency source code leaves a cascading impact for vulnerability disclosure for coins and tokens that share the same parent source code fork. We discuss both tactical and strategic recommendations for these security implications in Section 3.8.

3.7.2 Social Implications of Censorship

In this section we highlight the potentially profound social impacts of cryptocurrency censorship with two examples.

Limitation to Anonymous Funding Streams

Cryptocurrencies are used as anonymous funding streams for various organisations and individuals.

Firstly, we note that Bitcoin donations are a funding stream for the whistleblowing organisation WikiLeaks, as well as the investigative reporting platform the Organised Crime and Corruption Reporting Project [13, 15]. Cryptocurrencies are also used to anonymously fund political dissidents such as the late Alexei Navalny, an open critic of Vladimir Putin. Since 2016, Navalny received anonymous donations totalling 591 BTC, representing approximately 3 million USD [34]. The Tor project also accepts cryptocurrency donations to support their anonymous browsing software [16].

The motivation to censor cryptocurrencies is also considered to limit the ability to anonymously fund terrorist organisations such as Al-Qassam Brigades and Daesh, and also to disrupt money laundering activities [159, 35, 138, 24][4].

If access to cryptocurrencies was censored, the funding streams for these entities would be disrupted. Hindering funding to investigative journalism, whistleblowing, and anonymity projects like Tor could limit the freedom of the press, thus undermining a key aspect of modern democracy. Furthermore, if political opponents to current governments have their funding streams limited, this undermines the democratic element of fair representation for political beliefs. In contrast, censoring cryptocurrencies may also mitigate terrorist funding sources and help disrupt money laundering activities. Therefore,

[4]Al-Qassam Brigades is considered a terrorist group by the USA, EU, New Zealand, Australia, and the UK.

the act of censoring these digital assets requires consideration because the consequences of regulation have a more significant impact to democratic ideals than the censorship of file sharing services.

Disrupting Fiat Currency Relief

In Venezuela, cryptocurrencies like Bitcoin provide an option for asset 'escape value' where the local currency is under threat from hyperinflation due to civil unrest [124]. Where government banking systems are prone to collapse, cryptocurrencies provide a conduit of asset transfer protecting citizens from eroding currency values. Censoring access to cryptocurrencies would undermine the ability for citizens to escape the impacts of hyperinflation.

If we recall the Trusted Computing Group's definition of trust, it emphasizes the expectation of a system behaving in a particular manner for a specific purpose. This sets a standard for system integrity. In the case of cryptocurrencies, this expectation aligns with the ideal of decentralisation. However, our research reveals a contrast: the actual centralisation inherent in bootstrapping mechanisms. This discrepancy is important given the potential societal impacts of trust in these systems.

Our research highlights a particular aspect of cryptocurrency networks, offering a critical view about their centralised dependencies. This measurement study provides important insights, particularly in understanding the implications for individuals who rely on cryptocurrencies for money transfer. Beyond the trust in the decentralised nature of cryptocurrencies, these individuals may also not be aware of the potential leakage of their peer information. Our research gives users more knowledge to help them make more informed decisions about their use of cryptocurrencies.

3.8 Recommendations

In this section, we provide both tactical and strategic recommendations to improve bootstrapping functionality based on the findings of our research.

3.8.1 Tactical Recommendations

Tactical recommendations unfortunately assume the continued use of censorship-prone bootstrapping methods. However, they mitigate obvious design issues until strategic changes can be implemented.

Firstly, the current code bases should be reviewed for any incorrect configurations. We see examples of hard-coded IPs being invalid and unreachable. We also see a lack of diversity in bootstrapping methods. Therefore, the status of hard-coded seeds should be periodically updated, and diverse bootstrapping options should be configured to prevent a single form of censorship limiting access.

The third tactical recommendation we make targets the cryptocurrencies that return a small static set of IP addresses to rendezvous servers rather than returning a large dynamic set of peer IPs. We suggest core developers consider returning peer IPs in DNS query responses rather than the IPs of centralised servers. This recommendation requires consideration as it limits centralisation; however it introduces a side-channel to harvest peer IPs.

The fourth recommendation is to ensure that DNS seeding or IP hard-coding is never persistently used post-bootstrapping for peer reconnection. Peer caching should be used to limit the ongoing requirement on DNS for reconnection. Our final recommendation is to ensure that DNS seeds are diversified to at least two DNS providers to mitigate any outages that may occur to a single provider.

3.8.2 Strategic Recommendations

Strategic recommendations have overheads for implementation, performance, and protocol complexity. However, used in combination, they offer robust mitigations to the unwanted consequences of censorship-prone bootstrapping methods. We suggest these recommendations be included, not as default options, but as easily configurable fallback methods.

Our first strategic recommendation covers the use of DNS seeding as a bootstrapping option. Unfortunately, this recommendation still depends on DNS seeding, but addresses its shortcomings. DNS over TLS (DoT)/DNS over HTTPS (DoH), and DNSSEC provide channel encryption and query response integrity respectively. DoH has the added advantage of mitigating the identification of peer-to-peer bootstrapping traffic, as it runs on TCP port 443. DNSSEC would address the DNS manipulation that we see in Section 3.5. DoT or DoH would also ensure that the traffic was TCP based, thus removing the risk of UDP based DNS leakage when using Tor. However, this recommendation poses a large computational overhead to protocol operations due to the requirement to cryptographically sign a large dynamic set of IPs when using DNSSEC.

Our second recommendation would be creating GUI based options for Tor bootstrapping, similar to what Zcoin offers, and also offer Tor pluggable transport connectivity. This would allow peers the option to have obfuscated access to cryptocurrency peer-to-peer networks. We saw in Section 3.4.3 that Tor bootstrapping can be a realistic method for connection, but the technical configuration required may make this option prohibitive for several users. As part of this recommendation we also suggest that Tor hidden service `.onion` addresses be offered for initial connectivity.

In conclusion, we note that in isolation, none of the tactical and strategic recommendations is able to provide a full remedy to the censorship challenges inherent in peer-to-peer bootstrapping and address the social and security implications they cause. However, we know from numerous measurement surveys of Internet censorship techniques that the methods employed to block specific categories of traffic are not always effective [21, 162]. Therefore, our measurement study has highlighted that by offering multiple options for bootstrapping, a defense-in-depth approach can be adopted, thus providing resilience for this element of cryptocurrency peer-to-peer functionality. Furthermore, we are optimistic that by displaying the extent of source code inheritance further research will consider this finding when exploring other cryptocurrency vulnerabilities and risk exposure.

3.9 Additional Information

3.9.1 Software Forks and Hard Forks

We provide further details about the hard forks and software forks noted in Figure 3.7. A description of Genesis Blocks and Merkle Roots is given to understand the nature of cryptocurrency hard forks and how they differ from software forks. We recall that mapping the software forks on the cryptocurrencies allows us to highlight the lack of variance between the underlying source code. This allows other security researchers to expand the scope of vulnerability disclosure for individual cryptocurrencies which share a source code base with other coins.

Hard forks of cryptocurrencies occur when a blockchain permanently diverges into two separate chains at a particular block number. Cryptocurrencies which are hard forks of a parent cryptocurrency must have the same Genesis Block which includes the initial Merkle Root of the parent [26]. The latter is a strict cryptographic requirement which ultimately relates to the inception point of the blockchain construction. We note that software forks have no strict requirements for cryptographic immutability. Instead, they are simply source code repository forks, most commonly seen on the GitHub platform. For a discussion on the different types of forks, please refer to [4].

Komodos Genesis Block and Merkle Root

Interestingly, we see that although Komodo is a software fork of Zcash, it seems to have created its Genesis Block prior to Zcash, and it is the only coin to have the property of front running its parent in this manner. In fact, Komodo has another interesting property not captured by the visual. Komodo opted to use the initial Merkle Root that Bitcoin used. A blockchain explorer actually indicates that because the initial Merkle Root for Komodos Genesis Block was Bitcoins, the timestamp is recorded as 2009-01-03. Therefore, we counted the subsequent block in the blockchain as its Genesis Block, which occurred on 2016-09-13. In Figure 3.8 we provide an extract from the chainparams.cpp files hosted on GitHub for Bitcoin, Bitcoin Cash and Komodo. We see that Bitcoin Cash is a hard fork of Bitcoin because the Genesis Block and Merkle Root are identical. However, we see that Komodo is not a hard fork of Bitcoin because only the Merkle Root is the same. The Bitcoin Genesis Block is made up from the double iterated SHA256 hash output of the original Merkle Root 0x4a5e1e4b...afdeda33b but is also concatenated with other fields such as the timestamp, nonce, and version. So, although Komodo shares the same Merkle Root as Bitcoin it does not share the same Genesis Block because the latter mentioned fields are not the same as those used by Bitcoin. Therefore, it is clear from the properties of hashing functions that when the different fields are concatenated onto the Bitcoin Merkle Root that the resulting Genesis Block for Komodo outputs a different hash value to Bitcoin.

3.9.2 Tables

This section provides two tables that cover the details of our research. In Table 3.4 we provide information about the tooling used to complete our research. In Table 3.5 we show the top 100 cryptocurrencies we exported in our selection process.

3.10 Conclusion

In this chapter we provided a comprehensive view of the status of modern peer-to-peer bootstrapping techniques for cryptocurrencies. The act of bootstrapping, the initial

Bitcoin

```
assert(consensus.hashGenesisBlock == uint256S
("0x000000000019d6689c085ae165831e934ff763ae46a2a6c172b3f1b60a8ce26f"));
assert(genesis.hashMerkleRoot == uint256S
("0x4a5e1e4baab89f3a32518a88c31bc87f618f76673e2cc77ab2127b7afdeda33b"));
```

Bitcoin Cash

```
assert(consensus.hashGenesisBlock == uint256S
("000000000019d6689c085ae165831e934ff763ae46a2a6c172b3f1b60a8ce26f"));
assert(genesis.hashMerkleRoot == uint256S
("4a5e1e4baab89f3a32518a88c31bc87f618f76673e2cc77ab2127b7afdeda33b"));
```

Komodo

```
assert(consensus.hashGenesisBlock == uint256S
("0x027e3758c3a65b12aa1046462b486d0a63bfa1beae327897f56c5cfb7daaae71"));
assert(genesis.hashMerkleRoot == uint256S
("0x4a5e1e4baab89f3a32518a88c31bc87f618f76673e2cc77ab2127b7afdeda33b"));
```

Figure 3.8: Genesis Blocks and Merkle Roots of Bitcoin, Bitcoin Cash and Komodo.

Table 3.4: Tooling Used for Research. We note that the version given for RIPE Atlas, is based on the authors personal RIPE Atlas probe firmware version. †indicates Ubuntu's Uncomplicated Firewall on version 16.04 of the OS. ‡indicates Windows Defender Firewall on version 10 of the OS. The term 'core ref.' shorthand for core reference client software.

Tooling	**Category**	**Version**	**Usage**
Baretail	logfile viewer	3.50a	viewing debug / connection logs
Notepad++	text editor	7.5.8	development, log file viewing
MySQL	relational database	5.7.23	processing RIPE Atlas JSON data
Python	programming language	3.5.2	processing RIPE Atlas JSON data
RIPE Atlas	measurement platform	4940	collect DNS response data
TCPView	network activity monitor	3.05	view peer-to-peer connections
Tor	SOCKS proxy server	0.3.3.7	testing Tor bootstrapping
UFW†	software firewall	16.04	test censorship of each core ref.
WDF‡	software firewall	10	test censorship of each core ref.
Wireshark	network traffic debugger	2.6.2	debug network traffic of core ref.

Table 3.5: Top 100 Cryptocurrencies by Market Capitalization. Symbols: * = tokens transitioning to native 'mainnet' blockchains †= Omni token ‡= NEO token unmarked tokens = ERC20 tokens ⋆ = mineable ERC20 token for illustration.[10].

	non-mineable	**mineable**
coins	Ripple, NEO, Cardano, Stellar, IOTA, NEM, Qtum, Lisk, Nano, Steem, Stratis, Waves, BitShares, Decred, Ardor, Ark, PIVX, Factom, Byteball Bytes, ReddCoin, GXShares, Neblio, Nxt, Particl, Blocknet	Bitcoin, Ethereum, Bitcoin Cash, Litecoin, Dash, Monero, Ethereum Classic, Bitcoin Gold, Zcash, Bytecoin, Verge, Dogecoin, Siacoin, Electroneum, HShare, Zclassic, Komodo, Syscoin, Digibyte, Cryptonex, Monacoin, Bitcore, Zcoin, Vertcoin, SmartCash
tokens	EOS*, TRON*, VeChain*, Tether†, OmiseGO, ICON*, DigixDAO, Binance Coin, Populous, RChain, Maker, Status, Aeternity, Waltonchain, Augur, 0x, Veritaseum, Revain, Gas‡, KuCoin Shares, Basic Attention Token, Bytom, Zilliqa, Ethos, Loopring, Dragonchain, Golem, Nubulas, QASH, aelf, Polymath, Aion, Dent, Kyber Network, ChainLink, Dentacoin, IOStoken, FunFair, SALT, Kin, Power Ledger, Bancor, Enigma, Pillar, Request Network, Cindicator, MaidSafeCoin, TenX, Quantstamp, SinularityNET	0xBitcoin⋆

connection of a new peer to a distributed network, serves as the foundational step for the seamless operation of these decentralised digital assets.

This chapter measured the techniques employed by a diverse array of 74 cryptocurrencies, where we observed the prevalence of censorship-prone methodologies, such as DNS seeding and IP hard-coding. These mechanisms, while effective to some extent, expose vulnerabilities that required illumination so that mitigation techniques could be explored.

However, our study did not merely highlight vulnerabilities; it also illuminated pathways toward enhanced security and censorship-resistance. By embracing alternatives like Tor and ZMap, we unearthed promising avenues for circumventing censorship and fortifying the integrity of peer-to-peer connections.

Our global measurement study, powered by the distributed RIPE Atlas network, provided valuable perspectives into the network topologies and operational aspects of these cryptocurrencies. Alongside this, it revealed instances of disruption and manipulation, underscoring the need for robust and resilient bootstrapping mechanisms which required a different approach to the underlying dependence on centralised resources.

Our investigation unveiled a shared lineage of source code, underscoring the persistent influence of legacy bootstrapping techniques across diverse cryptocurrencies. This inheritance not only sheds light on historical design choices but also hints at a promising opportunity for cross-pollination of ideas and innovations motivated by the desire to reconsider the centralised dependencies still prevalent in these modern peer-to-peer networks.

Returning to the TCG definition of trust, our research demonstrated that although cryptocurrencies are expected to behave in a decentralised peer-to-peer manner, we identified that there are still essential elements of functionality dependent on centralised resources. Therefore, our measurement study clearly shows that we should question how we trust that cryptocurrencies behave in an expected manner for the specific purpose of trading and acquiring these digital assets.

3.10.1 Research Outlook

In this concluding section, we present a forward-looking perspective on prospective research paths within the domain of contemporary peer-to-peer networks.

While our focus within this chapter has been the scrutiny of cryptocurrency bootstrapping methods, the subject of centralised dependence within modern peer-to-peer networks extends beyond this single element. Other research avenues exist to explore other elements of reliance on centralised resources in these systems.

An area of research could be upon the exchanges that facilitate the acquisition and trading of cryptocurrencies—a practice that, paradoxically, straddles the principles of decentralisation and centralisation. Cryptocurrency exchanges have emerged as centralised hubs which are at odds with the original ideals of peer-to-peer transactions. Research in this domain can unravel the extent of dependence on these resources and potential methods to mitigate their use.

Moreover, source code repositories, serving as repositories of the foundational building blocks of cryptocurrencies, constitute yet another arena where a centralised dependency exists. The acquisition of GitHub by Microsoft has injected new considerations into the realm of open-source collaboration and dependence on a centralised resource. Exploring the implications of this centralisation and the potential impacts to the decentralised nature of cryptocurrencies offer another area of research in peer-to-peer networks.

Part 2: Implementations of Delay-Based Cryptography

In Part 2 of this thesis we focus our research on two original research projects in delay-based cryptography with a focus on practical implementations and applications.

Part 2 begins with Chapter 4 entitled ‘Foundations of Delay-Based Cryptography’ where we provide the historical context of delay-based cryptography and the necessary number theory required to properly explore the subject matter. This chapter provides crucial context for understanding the motivations behind the development of delay-based cryptographic protocols and their significance in modern cryptographic research.

Following the background exploration, Part 2 proceeds with the first original research contribution presented in Chapter 5, titled ‘Applications of Timed-release Encryption with Implicit Authentication’. In this chapter, we create a new property termed implicit authentication (IA) within a timed-release encryption (TRE) scheme. Our research motivation stems from the necessity to enhance the security and authenticity of encrypted communications in sensitive contexts such as whistleblowing. By integrating implicit authentication into TRE, we aim to fortify the integrity of encrypted disclosures and provide whistleblowers with an additional layer of protection against unauthorised tampering or manipulation of the data that they are disclosing.

This chapter represents an original contribution to the field of delay-based cryptography, offering a practical solution to address the challenges of securing time-delayed disclosures in adversarial environments.

Moving forward, Part 2 continues with Chapter 6, titled ‘TIDE: A Novel Approach to Constructing Timed-Release Encryption’. Here, we introduce a practical construction called TIDE (TIme Delayed Encryption), specifically designed for use in sealed-bid auctions. Building upon existing research on TRE, we address the challenges inherent in Vickrey and general sealed-bid auctions by devising a scheme that ensures the delayed release of encrypted information while facilitating the appropriate auction protocols. Through a detailed security analysis and implementation study, we demonstrate the efficacy and feasibility of TIDE in practical auction scenarios.

Part 2 concludes with Chapter 7 entitled 'Outlook of Delay-Based Cryptography'. In the final chapter of our thesis we explore the future prospects and potential applications of delay-based cryptographic schemes. Specifically, we consider the feasibility of leveraging TIDE as a component of a public randomness beacon. By extending the capabilities of TIDE we aim to add our insights into addressing the challenges in the construction of a public source of randomness.

In summary, Part 2 of this thesis focuses on the practical implementations and applications of delay-based cryptography. In both original research contributions we create practical and performance test concrete solutions and publish our results.

Chapter 4

Foundations of Delay-Based Cryptography

In this bridging chapter, we present the notion of timed-release encryption as a cryptographic primitive rooted in the concept of introducing delays. We consider the history of timed-release encryption to shed light on the motivation behind its emergence.

The concept of delay-based cryptography has emerged as a tool for various applications, ranging from securing auctions to ensuring the integrity of time-sensitive transactions. Delay-based cryptography involves associating standard 'wall-clock time' with an iterated sequential computation, thereby introducing a delay in the processing of a specific calculation. This delay can be used to achieve a variety of security goals, such as encrypting a message to be revealed only after a certain amount of time has elapsed or proving the passage of time in a verifiable manner.

A notable application of delay-based cryptography is in the construction of timed-release encryption (TRE) schemes. These schemes are designed to enable the secure distribution of sensitive information with a built-in time delay, ensuring that the information remains encrypted until a specified point in the future. This can be particularly valuable in scenarios such as whistleblowing, where individuals may need to disclose information gradually over time to protect themselves from retaliation.

In Part 2 of our thesis, we introduce a two new constructions of timed-release encryption schemes, which we term TRE-IA (Timed-Release Encryption with Implicit Authentication) and TIDE (TImed Delay Encryption) which build upon the foundational principles of delay-based cryptography. Unlike traditional TRE schemes that rely on time-servers or complex cryptographic primitives, TIDE utilises classic number-theoretic techniques to enable the decryption of encrypted messages after a specified delay period.

Our research focuses on addressing the practical challenges inherent in existing TRE schemes, particularly in the context of an encryption scheme for whistleblowers and for sealed-bid auctions. By seamlessly integrating RSA-OEAP encryption with time-lock puzzles, our constructions offers a practical and easily implementable solutions. Furthermore, we provide a thorough analysis of the security and efficiency of our constructions, demonstrating their effectiveness in real-world scenarios through implementation studies on consumer-grade hardware.

In the remainder of this bridging chapter, we provide the necessary background in foundational delay-based cryptographic primitives and the number theory underpinning the correctness and security of both our TRE-IA and TIDE constructions.

4.1 Use of This Chapter

In this bridging chapter we will also provide the preliminary material which will be common to the chapters in Part 2 of this thesis. This will include any assumptions, definitions, theorems, and algorithms which are common to the delay-based cryptography constructions in this part of the thesis. Moreover, the concluding part of this transitional chapter offers an overview of the correctness of our Part 2 timed-release encryption constructions, which are derived from the preliminary material introduced. The treatment of this bridging chapter aims to provide the reader with a valuable and easily accessible reference point for the constructions found in Part 2.

4.2 Introduction: Timed-Release Encryption

Timed-release cryptography, introduced as the concept of encrypting messages 'to the future' by May in 1993 within the Cypherpunks mailing list [123], has evolved into a pivotal element in modern cryptography. This approach, initially proposed for applications like delaying money transactions or transmitting messages in posthumous scenarios, has since been extensively refined and adapted across the cryptographic landscape. Notably, the significance of timed-release encryption extends beyond isolated use cases, finding relevance in a multitude of cryptographic contexts where enhanced security and privacy are imperative.

The first formal treatment of this subject was provided in 1996 by Rivest, Shamir and Wagner in their seminal work 'Time lock puzzles and timed-release crypto' [151]. In this work, the authors suggested encrypting a message in such a way that decrypting the message requires computing an iterated sequential function. The underlying cryptographic concept is that this computation must take at least a certain amount of real-world clock time to compute. This assumption is based upon the fact that each iteration of the function requires the input of the previous step, and hence one cannot run all of the steps in parallel, arbitrarily speeding up the computation. This seemingly simple idea birthed the rich subject of delay-based cryptography, which has found use in many different areas of cryptography.

4.2.1 Time-lock puzzles and Time-lock encryption

The foundation of delay-based cryptography is rooted in time-lock puzzles (TLPs), introduced by Rivest et al. in 1996 [151]. These cryptographic constructs encapsulate the principle of introducing delays through sequential computations, which resists parallelism and ensures a defined passage of time. The subsequent exploration of TLPs unveils the utilisation of RSA-based methods, where sequential computations based on modular arithmetic yield predictable delays. The prevalence of these techniques becomes evident as they form the cornerstone of constructions within Part 2 of this thesis.

In a TLP, an encryptor takes as input a string s and a time parameter t, and outputs a puzzle Z. The decryptor then spends t time running a sequential computation on the puzzle Z to re-construct the string s. In the original construction of Rivest et al., the method for obtaining a time-delay is repeated squaring in an RSA group. Explicitly, the encryptor samples an RSA modulus $N = pq$, where p and q are large primes, and chooses a string s, which they suggested could be a key to a symmetric encryption scheme. The encryptor then randomly samples r and computes the puzzle $Z = s + r^{2^t} \pmod{N}$.

The solver is then given Z and r, allowing for the computation of $r^{2^t} \bmod N$ in t sequential steps, and hence the solver can learn the string s. Note that using the trapdoor $\phi(N) = (p-1)(q-1)$, which requires knowledge of the factors of N, the encryptor can also construct the puzzle significantly faster than the solver can recompute it. This feature will be discussed extensively in Section 4.3.3. Whilst this is not the only method of constructing a cryptographic delay, it is certainly the most popular, and it is this construction that underpins our constructions in Part 2 of our thesis.

4.2.2 Timed-release encryption

The evolution of timed-release encryption intersects with modern cryptographic concepts, particularly public-key encryption (PKE), augmented with delay-based cryptographic mechanisms. This fusion presents an approach to safeguard sensitive information by allowing for its decryption only after a predetermined delay. Notably, recent advancements, as showcased by Chvojka et al. in 2021 [62], leverage TLP solutions within PKE frameworks to enable timed-release encryption. Part 2 of this thesis builds upon this foundation, considering the practical implementations and applications of timed-release encryption.

The earliest concept of timed-release encryption (TRE) was first mentioned by May in 1993 [123], with the idea of sending a message and a release time to a trusted agent, who would transfer the message at this release time. Modern instantiations of timed-release encryption are seen as a combination of public-key encryption (PKE) schemes with an additional aspect of delay [62, 112, 113]. The delay is achieved by encoding the decryption key as the solution to a sequential computation.

In 2021, Chvojka et al. introduced the idea of taking a TLP and using its solution in the key generation of a PKE scheme [62]. They use this to formally define a timed-release encryption (TRE) scheme where multiple parties encrypt a message to the public key of the PKE scheme. Then upon solving the puzzle they can reconstruct the secret key and decrypt all of the messages. The authors explain how to achieve this generically using standard TLP and PKE primitives.

The constructions in Part 2 of this thesis build on the timed-release encryption scheme formalised by Chvojka et al.

4.3 Preliminaries

In this section we provide preliminary material which will be common to the chapters in Part 2 of this thesis. This will include any assumptions, definitions, theorems, and algorithms which are common to the delay-based cryptography constructions in this part of the thesis.

4.3.1 Notation

We now provide the common notation for Part 2 of this thesis.

- $\mathbb{Z}_N^*$ – is the multiplicative group of integers modulo N [84].

- Blum Integer – is a special class of RSA modulus which is the product of two Gaussian primes i.e., $N = pq$, where $p \equiv q \equiv 3 \bmod 4$ [44].
- $\mathsf{negl}(\lambda)$ – is a negligible function in security parameter λ.
- $\mathsf{poly}(\lambda)$ – is a polynomial algorithm in security parameter λ.
- $\mathsf{Pr}[\,]$ – is the probability of an event.
- $\phi(N) = (p-1)(q-1)$ – is the Euler phi function which is the group order of $\mathbb{Z}_N^*$.
- $\mathcal{J}_N(x)$ – is the Jacobi symbol of N, which will evaluate to -1 or $+1$ [99].
- $\mathcal{O}^{\mathsf{Enc}}$ – an encryption oracle.
- $\mathcal{QR}_N$ – are the quadratic residues of N.
- $\mathcal{QNR}_N^{+1}$ – are the quadratic non-residues of N with positive a Jacobi symbol.
- $\mathcal{QNR}_N^{-1}$ – are the quadratic non-residue with a negative Jacobi symbol.
- $\simeq$ – is an isomorphism.

In our algorithms we use the following notation.

- $\leftarrow$ indicates a deterministic algorithm output.
- $\leftarrow_{\mathrm{R}}$ indicates a probabilistic algorithm output.
- $:=$ indicates assignment.
- $=$ indicates equality.
- $\neq$ indicates inequality.
- $()$ indicates a tuple.
- $\{\}$ indicates a set.
- $|x|$ indicates the cardinality of element x.
- $||$ indicates concatenation.
- $\in$ indicates inclusion.

- // denotes a comment.
- $\wedge$ indicates logical conjunction (and).
- $\vee$ indicates logical disjunction (or).
- $\neg$ indicates logical not.
- $\oplus$ indicates an exclusive or.
- $\lfloor x \rfloor$ indicates the floor of a real number x.
- $\mathsf{bin}(b)$ – is the binary representation of an integer b.
- $\mathsf{gcd}(a, N)$ – is the greatest common divisor of the positive integers a and N.
- $\mathsf{LSB}(x_i)$ – takes the Least Significant Bit of the positive integer x_i.
- $\mathsf{prime}(j)$ – is the Miller-Rabin algorithm which outputs a random j-bit Gaussian prime [126].
- $\mathsf{params}(1^k)$ – is a function which outputs the parameters for the RSA-OAEP PKE scheme [37].
- $\mathsf{rand}(k)$ – is a function that outputs a random k-bit integer.
- $\mathcal{U}(a, b)$ – uniformly selects of an integer that is between $a, b \in \mathbb{Z}$, where $a < b$ and a, b are inclusive.

4.3.2 Acronyms

We now provide the common acronyms for Part 2 of this thesis.

- BBS CSPRNG – Blum Blum Shub Cryptographically Secure Pseudo Random Number Generator [44]
- CPU – Central Processing Unit
- CRT – Chinese Remainder Theorem
- cVDF – Continuous Verifiable Delay Function
- DE – Delay Encryption

- DSS – Digital Signature Scheme
- EEA – Extended Euclidean Algorithm (calculates the greatest common divisor of integers a and N)
- EDF – Empirical Distribution Function
- GHz – Gigahertz (the clock speed of a computer processor)
- IBE – Identity-Based Encryption
- IND-CPA – Indistinguishability under Chosen-Plaintext Attacks
- IO – Indistinguishability Obfuscation
- IETF – Internet Engineering Task Force
- LQR – Law of Quadratic Reciprocity
- NIST – National Institute of Standards and Technology
- OS – Operating System
- PKCS – Public-Key Cryptography Standards
- PGP – Pretty Good Privacy
- PKE – Public-Key Encryption Scheme
- PPT - Probabilistic Polynomial Time
- PQC – Post Quantum Cryptography
- RAM – Random Access Memory
- RFC – Request For Comments (IETF Standards)
- RSW – Rivest Shamir Wagner
- RSA OAEP – Rivest Shamir Adleman Optimal Asymmetric Encryption Padding
- TIDE – TImed Delay Encryption
- TiB – Tebibyte (2^{40} bytes)
- TLP – Time-Lock Puzzle

- Tor – The Onion Router
- TRE – Timed-Release Encryption
- TRE-IA - TRE with Implicit Authentication
- VDF – Verifiable Delay Function

4.3.3 Importance of the RSW Time-Lock Assumption

We begin by introducing the formal definition of the Rivest Shamir Wagner (RSW) time-lock assumption [151] which is used in various cryptographic delay-based primitives including our timed-release encryption with implicit authentication (TRE-IA) and timed delay encryption (TIDE) constructions.

Definition 1. RSW time-lock assumption: Let $N = pq$ where p and q are distinct odd primes. Uniformly select $x \in \mathbb{Z}_N^*$, where $\mathbb{Z}_N^* = \{x \,|\, x \in (0, N) \wedge \mathsf{gcd}(x, N) = 1\}$. Then set the seed term as $x_0 := x^2 \bmod N$. If a PPT adversary $\mathcal{A}$ does not know the factorisation of N or group order $\phi(N) = (p-1)(q-1)$ then calculating $x_t \equiv x_0^{2^t} \bmod N$ is a non-parallelisable calculation that will require t sequential modular exponentiations calculated with the Algorithm 1 Square and Multiply [151].

Algorithm 1: Square and Multiply [66]

```
input  : $(a, b, N)$,  // $a, b, N \in \mathbb{N}$, $a^b \bmod N$
1 $d := 1$
2 $B := \mathsf{bin}(b)$  // $b$ in binary
3 for $j \in B$ do
4     $d := d^2 \bmod N$
5     if $j = 1$ then
6         $d := da \bmod N$
7     end
8 end
output: $d$
```

Distributed Generation of an RSA Modulus. For any delay-based construction that relies on the RSW time-lock assumption, the generation of a suitable group is pivotal for the execution of the repeated squaring and reduction process. The imperative is that the party orchestrating the delay computation must lack knowledge of any trapdoor that could accelerate the process.

The most prevalent avenue involves employing an RSA group, where the trapdoor is embodied by the Euler phi function $\phi(N) = (p-1)(q-1)$, given a composite $N = pq$. This RSA modulus, pivotal in time-lock puzzles and timed-release encryption, is typically generated in two distinct ways. One method necessitates the involvement of a centralised entity that generates and shares the modulus among solving parties for delay computation [151, 62]. This method is employed by the construction in our TIDE chapter to ensure the practicality of the implementation.

An alternative path involves gathering a collective of randomly selected or potentially anonymous entities with the setup. This scenario entails an efficiency versus central dependency trade-off. To maintain a desired efficiency level, the participant count must remain manageable, yet this also opens the door for collaborative breaches of construction security.

This approach mandates an intricate multi-party computation (MPC) ceremony, a field that has undergone significant advancement in recent years. In 2018, Frederiksen et al. [78] introduced an implementation for the malicious two-party setting, achieving an average runtime of 35 seconds using high-grade hardware and a 40.0 Gbps network link. A similar year saw Hazay et al. [94] propose a method leveraging threshold encryption for RSA modulus computation, demonstrating average CPU times of 15 minutes.

In 2020, Chen et al. [60] refined the multi-party protocol for biprime RSA modulus generation, addressing security concerns in prior models. Their work removed vulnerabilities present in earlier approaches and streamlined security assumptions. The trajectory of advancement continued in 2021, as Chen et al. [61] introduced Diogenes, an implementation of multi-party RSA modulus generation supported by thousands of parties. The architecture exhibited logarithmic communication growth per-party and withstood malicious attacks. Notably, the protocol executed with impressive efficiency, generating a 2048-bit modulus among 1,000 parties in under 6 minutes using their passive protocol, and under 25 minutes using the active variant.

However, the strength of this construction also bears a vulnerability; adversaries can execute a denial-of-service attack by compromising a single party. Although cheaters can be purged and the protocol restarted, adversaries with considerable influence can perpetuate such attacks and significantly delay modulus generation. This inherent vulnerability underlines the practical challenges still confronting the implementation of this approach, despite its recent feasibility.

In summation, while considerable strides have been taken to make this approach practicable, real-world implementation remains beset by substantial challenges. While our constructions in Part 2 of this thesis do not currently employ an MPC-generated RSA modulus, as this approach matures, a feasible retrospective adoption becomes increasingly viable.

Origins of the RSW Time-lock Assumption. The origins of the RSW time-lock assumption were derived from the BBS CSPRNG. Knowledge of the BBS CSPRNG will allow us to prove the correctness of the Square and Multiply algorithm specific to our TRE-IA and TIDE constructions.

Definition 2. Blum Blum Shub CSPRNG: The BBS CSPRNG generates cryptographically secure random numbers by iteratively extracting the least significant bit from the term x_{i+1} in the equation:

$$x_{i+1} = {x_i}^2 \mod N \tag{4.1}$$

where N is a Blum integer.

A Blum integer is a special class of RSA modulus which ensures that the cycle length of the BBS CSPRNG is long [44]. The security of the BBS CSPRNG is based on the difficulty of factoring the large composite Blum integer N which is the product of two large Gaussian prime numbers p and q, where $p \equiv q \equiv 3 \bmod N$. As well as providing assurance of long cycles lengths in the BBS CSPRNG, the modulus being a Blum integer is also a key requirement for the correctness of our schemes in Part 2. The properties relating to the correctness of our schemes will be discussed in Section 4.4.

We show the BBS CSPRNG pseudo code in Algorithm 2 which will generate a random binary digit s. Algorithm 2 starts by selecting $x \in \mathbb{Z}_N^*$ and calculates the seed value $x_0 \equiv x^2 \bmod N$. Next s is set to the binary digit 1 and the initial term x_i is set to the seed x_0. While the binary digit s is less than t bits the term x_{i+1} is calculated using the Square and Multiply Algorithm 1 with the input $(x_i, 2, N)$. Next, the least significant bit is extracted from x_{i+1} and concatenated to the binary digit s and then the term x_i is set to x_{i+1}. The loop terminates when s is a t bit binary digit.

The first t terms of a BBS CSPRNG sequence can be seen in Table 4.1, with their equivalent representations. On line 7 of Algorithm 2 the least significant bit is extracted from each term in a given row on Table 4.1.

Algorithm 2: Blum Blum Shub CSPRNG

```
   input : $(N, t)$, // $N$ is a Blum integer
1  $x := \mathcal{U}(2, N-1)$ // $x \in \mathbb{Z}_N^*$
2  $x_0 := x^2 \bmod N$
3  $s := 1$ // $s$ is a binary digit
4  $x_i := x_0$
5  while $|s| < t$ do
6      $x_{i+1} := {x_i}^2 \bmod N$ // Square and Multiply: input $(x_i, 2, N)$
7      $d := \mathsf{LSB}(x_{i+1})$
8      $s||d$
9      $x_i := x_{i+1}$
10 end
   output: $s$
```

Table 4.1: The first t terms of the BBS CSPRNG. The first row identifies which $i \in (1, \ldots, t)$ is being calculated, where 0 is the seed term. The second, third, and fourth rows are equivalent representations of the same term, i.e., in the penultimate column, for $i := t-1$, the terms x_{t-1}, x_{t-2}^2, and $x^{2^{t-1}}$ are equivalent.

i	0	1	2	...	$t-2$	$t-1$	t
x_i	x_0	x_1	x_2	...	x_{t-2}	x_{t-1}	x_t
x_{i-1}^2	x_0	x_0^2	x_1^2	...	x_{t-3}^2	x_{t-2}^2	x_{t-1}^2
$x_0^{2^i}$	x_0	$x_0^{2^1}$	$x_0^{2^2}$	...	$x_0^{2^{t-2}}$	$x_0^{2^{t-1}}$	$x_0^{2^t}$

We now prove the correctness of the Square and Multiply algorithm in the context of the BBS CSPRNG which is the underlying primitive in the RSW time-lock assumption.

Theorem 1. Algorithm 1 Square and Multiply correctly calculates the x_i term of the BBS CSPRNG.

Proof. The input to calculate the term x_i of the BBS CSPRNG takes as input $(x_0, 2^i, N)$, where x_0 is the seed term, and N is a Blum integer. Consider the base case when $i := 1$. The algorithm proceeds as follows: d is set to 1 and the exponent $b := 2^1$ is set to the binary string $B = 10$. Next, the algorithm enters the for loop on the first iteration. On the first iteration j is the first digit of B, which is 1. Next $d := 1$ is squared to output 1. Then the first conditional if statement is met as $j = 1$, therefore $d := 1x_0 = x_0 \bmod N$, and the first iteration of the loop is done. On the second iteration j is the second digit of B, which is 0. Next, as d was set to x_0 on the first iteration d is now set to $x_0^2 \bmod N$ on the second iteration. The first conditional if statement is not met, and the loop terminates as the final digit of B was processed. The algorithm then returns $d := x_1 \equiv x_0^2 \equiv x_0^{2^1} \bmod N$, as required. Therefore, the base case is true.

By the inductive hypothesis we claim that for any $i := k$, the loop invariant of Algorithm 1 returns the term $x_0^{2^k} \bmod N$ after k iterations. Therefore after k iterations, where b was set to 2^{k+1}, Algorithm 1 will have $d := x_0^{2^k} \bmod N$, and j will be the final digit of $B := 10\ldots0$. For any k, the variable B will be a binary string starting with the digit 1 followed by a trail of k digits equal to 0. This means after the first iteration of the for loop all remaining $j \in B$ will be 0. Thus, at the $k+1$ iteration of the for loop d will be set to $x_k^2 \bmod N$, and by definition $x_k^2 \equiv x_{k+1} \equiv x_0^{2^{k+1}} \bmod N$. Finally, Algorithm 1 will terminate at the $k+1$ iteration as the final digit of B was processed, and the algorithm will return $d := x_0^{2^{k+1}} \bmod N$. $\square$

Next, we show if the group order of N is known, denoted $\phi(N) = (p-1)(q-1)$, then the RSW time-lock assumption in Definition 1 is bypassed with a single $\log_2 N$ binary operation. That is, by using the Fermat-Euler Theorem, the input parameter b in Algorithm 1 can be reduced modulo the group order $\phi(N)$.

Theorem 2. Fermat-Euler Theorem. Let N be an odd prime, or let $N = pq$, where p and q are distinct odd primes. If $\mathsf{gcd}(a, N) = 1$, then $a^{\phi(N)} \equiv 1 \bmod N$, where $\phi(N) = N - 1$ if N is prime or $\phi(N) = (p-1)(q-1)$ if $N = pq$.

Proof. The proof of the Fermat-Euler Theorem is well known and can be found in [84]. $\square$

Corollary 1. Let $x_0 \in \mathbb{Z}_N^*$. If the group order $\phi(N)$ is known, then calculating x_t such that $x_t \equiv x_0^{2^t} \bmod N$ can be done in $\log_2 N$ binary operations.

Proof. Let $x_t \equiv x_0^{2^t} \bmod N$. If the exponent 2^t is reduced mod $\phi(N)$ we have $2^t = \alpha\phi(N) + \beta$, where β is the remainder of 2^t after the $\phi(N)$ modular reduction. Then, by Theorem 2 we have $x_t \equiv x_0^{2^t} \equiv x_0^{2^t \bmod \phi(N)} \equiv x_0^{\alpha\phi(N)+\beta} \equiv x_0^{\phi(N)^\alpha} x_0^\beta \equiv 1^\alpha x_0^\beta \equiv x_0^\beta \bmod N$. The number of bits in β is $O(\log N)$, and β is input into line 2 of Algorithm 1.

$\square$

Knowledge of the Fermat-Euler theorem is important in the context of the RSW time-lock assumption. In Example 3 we show how Theorem 2 is used to reduce $t\mathsf{log}_2 N$ operations to a single $\mathsf{log}_2 N$ when calculating $x_t := {x_0}^{2^t} \bmod N$. For simplicity, we employ a 64-bit modulus in our toy example for increased readability.

Example 3. Let $p := 4068102959$ and $q := 3776890883$, where both are Gaussian primes. Then $\phi(N) := (p-1)(q-1) = 15364780969107428956$ and $N := pq = 15364780976952422797$. Let the seed value $x_0 := 260115104422641727$. Therefore, to calculate the term $x_t := {x_0}^{2^t} \bmod N$, where $t := 1000000$ the following would be input into the Square and Multiply Algorithm $(a, b, N) = (x_0, 2^t, N)$. If the Fermat-Euler Theorem were not used, the exponent $b := 2^t$ would be the binary digit noted on line 2 of the Square and Multiply Algorithm with be the digit 1 followed by $t := 1000000$ trailing 0's. Therefore, calculating x_t without using the modular reduction in the Fermat-Euler Theorem would take $t\mathsf{log}_2 N$ sequential operations – thus agreeing with the RSW time-lock assumption. However, if the exponent b we reduced by the group order $\phi(N)$, then $b := 2^t \bmod \phi(N)$.

Therefore, the term B on line 2 of the Square and Multiply algorithm would be the binary digit 1001111010000000111101111000101111110101000101010011000111000000, and only a single $\mathsf{log}_2 N = 64$ bit operation would be required to calculate the term $x_t := {x_0}^{2^t \bmod \phi(N)} \bmod N = 5657671992503962019$. Note that the binary digit is the term β in Corollary 1.

4.3.4 Overview of Quadratic Residues

In this section we review the properties of quadratic residues and their relationship to our constructions in Part 2 of this thesis.

The number theoretic properties of quadratic residues have been studied by prominent mathematicians such as Euclid, Euler, Gauss, and Fermat [84]. Euclid studied quadratic residues in the 3rd century BC and developed the first proof of the Law of Quadratic Reciprocity.

Simply stated, if p is an odd prime number, then the Law of Quadratic Reciprocity allows us to determine if the equation $x^2 \equiv r \mod p$ has a solution. In the 16th century, Leonhard Euler used quadratic residues to prove the theorem of quadratic reciprocity and to establish a number of fundamental results in number theory. By the 19th century, Carl Friedrich Gauss used quadratic residues to prove the Law of Quadratic Reciprocity and to make important discoveries about prime numbers. Quadratic residues are currently being employed in modern cryptography applications. In this thesis, we leverage the number theoretic properties of quadratic residues and incorporate them into delay-based cryptographic constructions. The correctness and soundness of our constructions in Part 2 rely on diverse properties of quadratic residues.

To provide context for this thesis, we review the relevant background information on quadratic residues.

Definition 3. Quadratic Residues in $\mathbb{Z}_N^*$ are numbers r that satisfy congruences of the form:

$$x^2 \equiv r \mod N \tag{4.2}$$

If an integer x exists such that the preceding congruence is satisfied, we say that r is a quadratic residue of N. If no such x exists such that the preceding congruence is satisfied, we say that r is a quadratic non-residue of N.

Referring to Section 4.3.3 and the RSW time-lock assumption, we can observe, by Definition 3, that every term x_{i+1} in the BBS CSPRNG, noted in Definition 2, is a quadratic residue. This is demonstrated in Equation 4.1, where each term x_{i+1} is the square of the preceding term x_i, i.e., ${x_i}^2 \equiv x_{i+1} \bmod N$.

Recognising that each term in both a BBS CSPRNG and thus an RSW time-lock puzzle is a quadratic residue, we can leverage some established number-theoretic principles to devise innovative delay-based cryptographic constructions such as TRE-IA and TIDE with simple and verifiable proofs of correctness.

To expand on quadratic residues and their relevance to our delay-based constructions, we offer a brief overview of Jacobi symbols.

The Jacobi symbol, denoted $\mathcal{J}_N(r)$, is a function which defines the quadratic character of r in Equation 4.2. The Jacobi Symbol can be calculated in polynomial time using Euler's Criterion.

Theorem 3. *Euler's Criterion can be used to calculate the Jacobi Symbol of the number r in Equation 4.2 for a prime modulus p. If $\mathsf{gcd}(r, p) = 1$, then:*

$$\mathcal{J}_p(r) = r^{\frac{p-1}{2}} = \begin{cases} +1, \text{if} r \in \mathcal{QR}_p \\ -1, \text{if} r \in \mathcal{QNR}_p \end{cases} \tag{4.3}$$

Where $r \in \mathcal{QR}_p$ indicates that r is a quadratic residue of p and $r \in \mathcal{QNR}_p$ indicates that r is a quadratic non-residue of p.

Proof. The proof of Theorem 3 is well known and can be found in [84]. □

When the modulus is a prime number if the Jacobi symbol evaluates to $+1$ then r is always a quadratic residue and if the Jacobi symbol evaluates to -1 then r is always a quadratic non-residue. The Jacobi symbol is more complex when the modulus is a composite number $N = pq$.

Corollary 2. (Of Theorem 3). Euler's Criterion can be used to calculate the Jacobi Symbol of the number r in Equation 4.2 for a composite modulus N in polynomial time if the factorisation of N is known.

Proof. The proof of Corollary 2 is well known and can be found in [84]. □

Algorithm 3 shows how to determine the quadratic character of r for composite N using Theorem 3 and Corollary 2. When N is composite the quadratic character of r can take three formats. If the Jacobi symbol evaluates to -1 then r is always a quadratic non-residue, denoted $\mathcal{QNR}_N^{-1}$. However, if the Jacobi symbol evaluates to $+1$ then r can either be a quadratic residue, denoted $\mathcal{QR}_N$ or a quadratic non-residue denoted $\mathcal{QNR}_N^{+1}$. In Algorithm 3 the output is x, where x is the quadratic character of r and will be equal to one element in the set $\{\mathcal{QR}_N, \mathcal{QNR}_N^{+1}, \mathcal{QNR}_N^{-1}\}$.

Algorithm 3: Calculating $\mathcal{J}_N(r)$ for composite N.

```
  input  : $(r, p, q)$
1 $\mathcal{J}_p(r) := r^{\frac{p-1}{2}} \bmod p$
2 $\mathcal{J}_q(r) := r^{\frac{q-1}{2}} \bmod q$
3 if $\mathcal{J}_p(r) = 1 \wedge \mathcal{J}_q(r) = 1$ then
4 |   $x := \mathcal{QR}_N$
5 else if $\mathcal{J}_p(r) = -1 \wedge \mathcal{J}_q(r) = -1$ then
6 |   $x := \mathcal{QNR}_N^{+1}$
7 else
8 |   $x := \mathcal{QNR}_N^{-1}$
9 end
  output: $x$
```

If the factorisation of N is unknown the Jacobi symbol of r can still be calculated. Algorithm 4 presents the algorithm to calculate the Jacobi symbol of r. The original Python code for the function $\mathsf{Jacobi}(x, \mathsf{N})$ can be found in [65].

In Algorithm 4 the output is $\mathcal{J}_N(r)$, where $\mathcal{J}_N(r)$ will be equal to one element in the set $\{-1, +1\}$. This differs slightly from Algorithm 3, where x will be equal to one element in the set $\{\mathcal{QR}_N, \mathcal{QNR}_N^{+1}, \mathcal{QNR}_N^{-1}\}$. This difference arises because when the factorisation of N is unknown, the Jacobi symbol can still be calculated. However, if the Jacobi symbol evaluates to $+1$ then the quadratic character of r will remain unknown.

Algorithm 4: Jacobi is run on random input r and modulus N to calculate the Jacobi symbol $\mathcal{J}_N(r)$.

```
input  : $r, N$
1  $\mathcal{J}_N(r) = 1$
2  while $r \neq 0$ do
3      while $r \bmod 2 = 0$ do
4          $r \coloneqq \frac{r}{2}$
5          if $N \bmod 8 = 3 \vee N \bmod 8 = 5$ then
6              $\mathcal{J}_N(r) \coloneqq -\mathcal{J}_N(r)$
7          end
8      end
9      if $r \bmod 4 = 3 \wedge N \bmod 4 = 3$ then
10         $\mathcal{J}_N(r) \coloneqq -\mathcal{J}_N(r)$
11     end
12     $r,\ N \coloneqq N,\ r$
13     $r \coloneqq r \bmod N$
14 end
15 if $N \neq 1$ then
16     $\mathcal{J}_N(r) \coloneqq 0$
17 end
output: $\mathcal{J}_N(r)$
```

That is, if the Jacobi symbol evaluates to -1 then r will definitely be a quadratic non-residue, denoted $\mathcal{QNR}_N^{-1}$. However, if the Jacobi symbol evaluates to $+1$ then r can be either element from the set $\{\mathcal{QR}_N, \mathcal{QNR}_N^{+1}\}$. The latter outcome of $\mathcal{J}_N(r)$ evaluating to $+1$ for composite modulus N forms the premise of the Quadratic Residuosity Problem [99].

Definition 4. Quadratic Residuosity Problem. Given integer r and $N = pq$, when p and q are unknown, if $\mathcal{J}_N(r)$ evaluates to $+1$ then determine if $r \in \mathcal{QR}_N$ or if $r \in \mathcal{QNR}_N^{+1}$.

Familiarity with Definition 4 will play a key role in our TIDE construction by aiding the selection of a seed value for the generation of a time-lock puzzle. Importantly, the understanding that the Jacobi symbol can be computed without necessitating awareness of the modulus factors contributes to diminishing the dependency on the entity responsible for setting up the challenge. In Section 6.3, we will explore the practical implementation of timed-release encryption using our innovative approach. Notably, the capability to calculate the Jacobi symbol without the need for knowledge of the modulus factors significantly reduces the reliance on a single entity when generating the time-lock puzzle, highlighting an advantage demonstrated by TIDE.

Next, we explore the distribution of quadratic residues and non-residues, as well as the Law of Quadratic Reciprocity. This knowledge will be crucial in demonstrating the correctness of our TRE-IA and TIDE constructions. Specifically, when dealing with composite modulus N, quadratic residues and non-residues exhibit a distinct distribution within the $\mathbb{Z}_N^*$.

Theorem 4. *The cardinality of $\mathcal{QR}_N$, $\mathcal{QNR}_N^{+1}$, and $\mathcal{QNR}_N^{-1}$ for composite $N = pq$, where p and q are distinct primes is as follows:*

$$\begin{aligned} |\mathcal{QR}_N| &= \frac{|\mathbb{Z}_N^*|}{4} = \frac{\phi(N)}{4}. \\ |\mathcal{QNR}_N^{+1}| &= \frac{|\mathbb{Z}_N^*|}{4} = \frac{\phi(N)}{4}. \\ |\mathcal{QNR}_N^{-1}| &= \frac{|\mathbb{Z}_N^*|}{2} = \frac{\phi(N)}{2}. \end{aligned} \tag{4.4}$$

Where, $|\mathbb{Z}_N^*| = \phi(N) = (p-1)(q-1)$, and $\phi(N)$ is Euler's totient function.

Proof. The proof of Theorem 4 is well known and can be found in [99]. □

Euler's Criterion in Theorem 3 and Corollary 2 tells us if r is a quadratic residue of prime N or composite modulus N. The Law of Quadratic Reciprocity is a theorem that provides a rule for determining whether or not a given integer is a quadratic residue modulo a prime number.

Theorem 5. Law of Quadratic Reciprocity. Let p and q be distinct odd coprime integers, then the Jacobi symbol is defined as:

$$\mathcal{J}_p(q) = \begin{cases} +1, \text{if} q \in \mathcal{QR}_p \\ -1, \text{if} q \in \mathcal{QNR}_p \end{cases} \tag{4.5}$$

Then:

$$\mathcal{J}_p(q)\mathcal{J}_q(p) = (-1)^{(\frac{p-1}{2})(\frac{q-1}{2})} = \begin{cases} +1, \text{if } p \equiv 1 \bmod 4 \text{ or } q \equiv 1 \bmod 4 \\ -1, \text{if } p \equiv 3 \bmod 4 \text{ and } q \equiv 3 \bmod 4 \end{cases} \tag{4.6}$$

Therefore, supported by Euler's Criterion, as stated in Theorem 3, :

$$\mathcal{J}_p(-1) = (-1)^{\frac{p-1}{2}} = \begin{cases} +1, \text{if } p \equiv 1 \bmod 4 \\ -1, \text{if } p \equiv 3 \bmod 4 \end{cases} \tag{4.7}$$

Proof. The proof of Theorem 5 is well known and can be found in [84]. □

4.3.5 Taking Square Roots Modulo N

In the previous sections in Part 2 we describe the importance of the RSW time-lock assumption and how it relates to the BBS CSPRNG. We also demonstrated how each term in an BBS CSPRNG sequence and thus each term in an RSW time-lock puzzle is a quadratic residue and provided some key properties of quadratic residues relevant to our Part 2 constructions.

We showed how calculating $x_t \equiv {x_0}^{2^t} \bmod N$ will take t sequential $\log_2 N$ computation if the factorisation of N is not known. We then showed that if the factorisation of N is known, then x_t can be calculated in a single $\log_2 N$ computation by calculating $x_t \equiv {x_0}^{2^t \bmod \phi(N)} \bmod N$.

In each of the examples above, we illustrated how to calculate terms in the typical *forward* direction of an RSW time-lock puzzle. That is, given the seed term x_0 calculating the t^{th} term x_t in a sequence is either a *moderately difficult* time-locked calculation if the factorisation of N is unknown or a single calculation if the group order of the modulus N can be calculated.

Informally, a moderately difficult calculation refers to a computational task that requires a reasonable amount of time and computational resources to solve. It is designed to provide a certain level of security and delay, ensuring that unlocking the puzzle or revealing the solution requires a significant effort.

In this section we first consider an opposing challenge of calculating terms in the *backward* direction of an RSW time-lock puzzle.

Challenge: Given the seed term x_0 and the final term x_t in a RSW time-lock puzzle how can the term $x_{t-1} \equiv \sqrt{x_t} \bmod N$ be calculated?

One method to find x_{t-1} is to calculate $x_{t-1} = \sqrt{x_t} \equiv {x_0}^{2^{t-1}} \bmod N$. However, Definition 1 tells us this is a moderately difficult computation requiring $t-1$ sequential steps if the factorisation of N is unknown. The question then remains – is there a faster method to move in the backward direction of a RSW time-lock puzzle if the factorisation of the modulus N is unknown? As it turns out, there is no faster computation to calculate $\sqrt{x_t} \bmod N$ because performing this calculation is equivalent to factoring N.

The proof that taking square roots modulo N is equivalent to factoring N is attributed to Michael Rabin, who introduced the concept in his paper 'Digitalized signatures and public key functions as intractable as factorization' published in 1979 [145].

Theorem 6. Let $N = pq$, where p and q are λ bit odd primes. Then given any $r \in \mathcal{QR}_N$, finding x such that $x^2 \equiv r \bmod N$ is equivalent to factoring N.

Proof. Proof is well known and can be found in [145]. □

This seminal result forms the basis for our TRE-IA and TIDE constructions in Part 2 of this thesis.

In our TRE-IA chapter, the solver is given the challenge $C \coloneqq (x_0, x_t)$ and must find the decryption key of an RSA-OAEP PKE system. The decryption key is the term x_{t-1} which is calculated using the seed x_0 as follows $x_{t-1} \equiv \sqrt{x_t} \equiv {x_0}^{2^{t-1}} \bmod N$. Therefore, if taking the square root of x_t was an easy calculation then there would be a contradiction to the RSW time-lock assumption.

In our TIDE chapter, the solver is given the challenge $C \coloneqq (x, x_0, x_{-t})$. In the challenge x is selected such that $x \in \mathcal{QNR}^{-1}$ and $x^2 = x_0 \bmod N$. In this construction the seed term is x_{-t} and moderately difficult challenge is to sequentially calculate

$(x_{-t})^{2^{t-1}} = x' = \sqrt{x_0} \bmod N$. We will prove in our TIDE chapter that x' is distinct from x and that this knowledge allows the factorisation of the modulus N. Once the factorisation of N is known it is then possible to recover the decryption key by calculating the group order $\phi(N) = (p-1)(q-1)$, and then using EEA to recover the multiplicative inverse of the encryption key e. That is d is recovered as $d \coloneqq e^{-1} \bmod \phi(N)$. Similar to our TRE-IA construction, if taking the square root of x_0 was an easy calculation then the RSW time-lock assumption would again be contradicted.

Finally, in our TIDE chapter, the creation of the challenge $C \coloneqq (x, x_0, x_{-t})$ requires taking square roots modulo N to move in the *backward* direction of a RSW time-lock puzzle. Although Theorem 6 states, taking square roots modulo N is equivalent to factoring, if the factors of N are known, then taking square roots is possible. Furthermore, with the use of Gaussian primes in our constructions, taking square roots modulo N is a straight forward calculation. This is done by utilising the Chinese Remainder Theorem and a result derived from Euler's Criterion in Theorem 3. We provide an overview of our method of taking square roots modulo N in Section 4.4.1 and then provide the full exposition into the detailed construction of TIDE in Chapter 6.

In the following section we will provide the remaining number theory to prove the correctness of our Part 2 constructions.

4.4 The Correctness of Our Constructions

This section introduces to the Chinese Remainder Theorem and its importance in guaranteeing the correctness of our constructions. Once we cover the Chinese Remainder Theorem, we will have the necessary number theory background to fully explore calculating the terms in the *backward* direction of an RSW time-lock puzzle. We will then show how the Chinese Remainder Theorem also creates both a challenge and an advantage to our constructions when considering the solutions to the square root of any $r \in \mathcal{QR}_N$.

Finally, we will provide an overview of the RSA-OAEP PKE scheme and provide a high level view of how we ensure the correctness in the selection of the encryption and decryption exponents in our Part 2 constructions.

4.4.1 The Chinese Remainder Theorem

The CRT will first provide us with the necessary number theory to show how we can move in the *backward* direction of an RSW time-lock puzzle. We will then show how the CRT implies that there are four solutions to the quadratic congruence in Equation 4.2.

Theorem 7. (**The Chinese Remainder Theorem**) Let the following be a system of linear congruences:
$y \equiv \alpha_1 \mod n_1$
$y \equiv \alpha_2 \mod n_2$
$\vdots$
$y \equiv \alpha_k \mod n_k$
Where $y, n_i \in \mathbb{Z}^+$, $\alpha_i \in \mathbb{Z}$, and α_i are arbitrary integers and each n_i is pairwise coprime $\forall i \in \{1, \ldots, k\}$. Then, this system of linear congruences is guaranteed to have a unique solution mod N:

$$y = \sum_{i=1}^{k} \alpha_i N_i N_i^{-1} \mod N \tag{4.8}$$

Where $N = \prod_{i=1}^{k} n_i$, $N_i = \frac{N}{n_i}$, and N_i^{-1} is the multiplicative inverse of N_i mod n_i, i.e., $N_i N_i^{-1} \equiv 1 \mod n_i$. We also recall that each N_i^{-1} can be found in polynomial time using the Extended Euclidean Algorithm.

Proof. Proof is well known and can be found in [99]. □

We can now resume the discussion in Section 4.3.5 about moving backward in an RSW time-lock puzzle sequence. With our knowledge of the Chinese Remainder Theorem, we can now demonstrate how this is accomplished when the factors of the modulus N are known. In our Part 2 constructions, we use a Blum integer as our modulus N, which is the product of two Gaussian primes satisfying $p \equiv q \equiv 3 \bmod 4$. This algebraic property of N enables us to easily compute the square roots of any quadratic residue modulo p and q by applying Euler's Criterion, as presented in Theorem 8. Once we have the square roots of a quadratic residue modulo p and q we can use the CRT to take square roots modulo N.

To begin, we will demonstrate how to take square roots modulo a Gaussian prime.

Theorem 8. Let p be a Gaussian prime. For any $r \in \mathbb{Z}_p^*$, if $\mathcal{J}_p(r) = +1$, then finding α such that $\alpha \equiv \sqrt{r} \bmod p$ can be found by calculating $\alpha \equiv r^{\frac{p+1}{4}} \bmod p$.

Proof. Let $\alpha = r^{\frac{p+1}{4}} \bmod p$. Then $\alpha^2 \equiv (r^{\frac{p+1}{4}})^2 \equiv r^{\frac{2p+2}{4}} \equiv r^{\frac{p+1}{2}} \bmod p$. Next, let $\frac{p+1}{2} = 1 + \frac{p-1}{2}$. Therefore, by Euler's Criterion (Theorem 3) $\alpha^2 \equiv r^1 r^{\frac{p-1}{2}} \equiv r \bmod p$. We refer to α as the principal square root of r mod p. $\square$

We demonstrate in Example 4 how we apply Theorem 8 and the Chinese Remainder Theorem, as presented in Theorem 7, to take square roots modulo a Blum integer N. This process enables us to move in the *backward* direction of an RSW time-lock puzzle sequence as first discussed in Section 4.3.5.

Example 4. Let $N = 67 \cdot 139 = pq = 9313$. Given the seed $x_0 = 776 \in \mathcal{QR}_N$, the square root of x_0 mod N, denoted by $x_{-1} = \sqrt{x_0}$, can be found as follows:

- calculate $\alpha \equiv x_0^{\frac{p+1}{4}} \equiv x_0^{17} \equiv 21 \bmod p$
- calculate $\beta \equiv x_0^{\frac{q+1}{4}} \equiv x_0^{35} \equiv 9 \bmod q$
- calculate $x_{-1} = \sqrt{x_0} = \alpha q(q^{-1} \bmod p) + \beta p(p^{-1} \bmod q) = 128862$

Then α and β are calculated using Theorem 8 and x_{-1} is calculated using the CRT. Note that $(q^{-1} \bmod p)$ and $(p^{-1} \bmod q)$ are calculated using Euclid's Extended Algorithm. To verify correctness, note that $128862^2 \equiv 776 \equiv x_0 \bmod N$.

In the remainder of this section we should how the Chinese Remainder Theorem implies there are four solutions to every quadratic congruence. Therefore, every quadratic residue in Equation 4.2 has precisely four unique square roots. That is, $\pm x \equiv \pm x' \equiv \sqrt{r} \bmod N$, where $r \in \mathcal{QR}_N$ and where $x \neq \pm x'$.

We begin by discussing the challenge that the CRT introduces to our TRE-IA construction. The Chinese Remainder Theorem implies that there are four distinct square roots for $\sqrt{r} \bmod N$. This poses a challenge as the solution to the RSW time-lock puzzle is the decryption key $d := \sqrt{x_t} \bmod N$. As the TRE-IA construction uses the RSA-OAEP Public-Key Encryption scheme as a basis for its functionality, it is necessary to prove that the decryption key is unique. In our TRE-IA chapter we will show how the Law of Quadratic Reciprocity in Theorem 5 provides us with the proof that the decryption key d is indeed unique.

Conversely, in our TIDE construction we use the four distinct square roots of $\sqrt{r}$ mod N to our advantage. In the TIDE construction the solver is initially given two out of four square roots for $\sqrt{r} \bmod N$. Then, the RSW time-lock puzzle is generated such that the solution reveals the other two square roots of r. Knowledge of all four square roots of r allows us to use Fermat's factorization method and the weaker congruence of squares condition to factor the modulus N.

We now consider the Chinese Remainder Theorem from a different perspective. That is, in Theorem 7, if we are given y and the moduli n_i, find the solutions for each α_i. When the CRT is considered in the latter manner an equivalent statement known as the Chinese Remainder Theorem Isomorphism is used. In this thesis we are concerned in applying the results of the CRT in the case where $N = pq$, where p and q are distinct odd primes. Therefore, we limit now limit our application of the CRT Isomorphism to this case.

Definition 5. The Chinese Remainder Theorem Isomorphism.
In the case of $N = pq$, where p and q are distinct odd primes, the Chinese Remainder Theorem Isomorphism is denoted by

$$\mathbb{Z}_N^* \simeq \mathbb{Z}_p^* \times \mathbb{Z}_q^* \tag{4.9}$$

Simply stated, each $y \in \mathbb{Z}_N^*$ is equivalent (isomorphic) to a tuple $([y \mod p], [y \mod q])$.

Therefore, Theorem 7 and Definition 5 demonstrate that if $r \in \mathcal{QR}_N$ then there are four distinct solutions to Equation 4.2.

Theorem 9. For all $N = pq$, where p and q are distinct odd primes, each $r \in \mathcal{QR}_N$ has four distinct solutions.

Proof. The CRT Isomorphism can be used to prove that there are four square roots modulo $N = pq$, where p and q are two distinct prime numbers in the following manner.

Let $x^2 \equiv r \mod N$ be a quadratic congruence, where $r \in \mathcal{QR}_N$. Using the CRT Isomorphism, we can write N as $N = pq$ and consider the two equations $x^2 \equiv r \mod p$ and $x^2 \equiv r \mod q$ separately.

Since p and q are prime, we know that $\mathbb{Z}_p$ and $\mathbb{Z}_q$ are fields. Therefore, each of these equations has exactly two solutions in their respective fields. To see why this is the case, note that $\mathbb{Z}_p$ is a field, which means that every non-zero element has an inverse. In particular, if $x^2 \equiv r \mod p$ has a solution x, then we can write $r \equiv x^2 \bmod p$ and divide both sides by x (since x is non-zero) to obtain $rx^{-1} \equiv x \bmod p$. This means that rx^{-1} is also a solution to the equation.

Conversely, if rx^{-1} is a solution to the equation, then we have $(rx^{-1})^2 \equiv r \bmod p$, which means that $x^2 \equiv r \bmod p$.

Thus, the equation $x^2 \equiv r \bmod p$ has either two solutions or no solutions in the field $\mathbb{Z}_p$ depending on whether r is a quadratic residue modulo p or not. Let these solutions be x_p and x_q, respectively.

By the CRT Isomorphism, we know that there exists a one-to-one correspondence between the solutions of $x^2 \equiv r \bmod p$ and $x^2 \equiv r \bmod q$ and the solutions of $x^2 \equiv r \bmod N$. This correspondence can be written as: $x \equiv x_p \bmod p$ and $x \equiv x_q \bmod q$.

By the Chinese Remainder Theorem, this system of linear congruences has exactly four solutions modulo N. Therefore, there are exactly four square roots modulo N when N is the product of two prime numbers. □

Now we show that if all four distinct solutions of Equation 4.2 are known, then we can factor N in polynomial time using Fermat's factorization method and the weaker congruence of squares condition.

4.4.2 Fermat's Factorisation Method

In this section we show how we use the four solutions to the modular congruence $\sqrt{r} \bmod N$, when $r \in \mathcal{QR}_N$, to factor N using Fermat's Factorisation Method and the weaker congruence of square condition.

In our TIDE construction in Chapter 6 the recovery of the decryption key d is solved by factoring the Blum integer N and then recovering d as the multiplicative inverse of the encryption exponent e. This is possible as the solution to an RSW time-lock puzzle in this construction recovers the required information to factor N. Then the decryption key d can be recovered as $d := e^{-1} \bmod (p-1)(q-1)$ using EEA. Therefore, in the TIDE construction, the recovery of the decryption key d is done *indirectly* by first solving the RSW time-lock puzzle and then factoring N using the congruence of squares condition.

Conversely, in Chapter 5 in our TRE-IA construction the recovery of the decryption key d is solved by computing the solution to an RSW time-lock puzzle directly. The challenge $C := (x_0, x_t)$ is given and the solver must *directly* find $d = x_{t-1} \equiv \sqrt{x_t} \equiv {x_0}^{2^{t-1}} \bmod N$ by running the Square and Multiply Algorithm noted in Algorithm 1. As the decryption key is directly found for the TRE-IA construction, the requirement to use the results from Fermat's Factorisation Method do not apply.

We now provide the definitions of Fermat's Factorisation Method and the weaker congruence of squares condition.

Definition 6. Fermat's Factorisation Method If integers x and x' are found such that they satisfy the equality $x^2 - x'^2 = N$, then N can be factored with the equation $x^2 - x'^2 = (x + x')(x - x') = N$.

Definition 7. Congruence of Squares Condition If integers x and x' are found such that they satisfy the congruence $x^2 \equiv x'^2 \bmod N$, where $x \neq \pm x'$, then $x^2 - x'^2 \equiv (x + x')(x - x') \equiv 0 \bmod N$. Therefore, $N \mid (x + x')(x - x')$ and $(x + x')$ and $(x - x')$ each contain the factors of N which can be recovered in polynomial time by calculating $\mathsf{gcd}(x + x', N)$ and $\mathsf{gcd}(x - x', N)$ using EEA.

In Definition 7 it is possible that calculating $\mathsf{gcd}(x + x', N)$ and $\mathsf{gcd}(x - x', N)$ may only yield the trivial factors 1 and N. However as the modulus N is typically 2048 bits or larger, the probability of this in practice is negligible. For intuition we provide a toy example of the congruence of square condition being used.

Example 5. Let N be the Blum integer $125249 = 251 \cdot 499$. To apply the congruence of squares method, we need to find two integers x and x' such that $x^2 \equiv x'^2 \bmod N$ but $x \neq \pm x'$. One such pair of values is $x = 100$ and $x' = 8383$. We thus have $100^2 \equiv 8383^2 \equiv 10000 \bmod 125249$, therefore $(8383 - 100) = 8283$ and $(8383 + 100) = 8483$. We can then recover the non-trivial factors $\mathsf{gcd}(8283, 125249) = 251$ and $\mathsf{gcd}(8483, 125249) = 499$.

To conclude our Part 2 preliminaries section we discuss the correctness of the encryption and decryption keys of our TRE-IA and TIDE constructions. Both constructions use a RSA-OAEP based PKE scheme with a RSW time-locked decryption exponent.

4.4.3 Overview of RSA-OAEP

In this section, we will provide an overview of the RSA-OEAP (Optimal Asymmetric Encryption Padding) PKE (Public-Key Encryption) scheme. Our Part 2 constructions rely on this scheme for the confidentiality of messages.

First, we will give a brief description of the scheme, followed by a discussion of how the scheme works in $\mathbb{Z}_N^*$ – the ring of integers modulo N. By explaining the properties of $\mathbb{Z}_N^*$ we aim to provide context to the correctness of our Part 2 constructions.

Each of our schemes uses a RSW time-lock for the decryption exponent of an RSA-OAEP based PKE scheme. However, each scheme generates the RSA-OEAP encryption and decryption exponents differently. Therefore, it is essential for us to demonstrate that the encryption and decryption exponents in our scheme align with the encryption and decryption functions of the RSA-OEAP scheme.

The RSA-OEAP is a public-key encryption scheme based on the original textbook RSA cryptosystem [149]. RSA-OAEP was introduced in 1994 by Bellare and Rogaway in their paper 'Optimal Asymmetric Encryption — How to Encrypt with RSA' [37]. It uses a random process to pad messages to avoid revealing information about the plaintext through the ciphertext.

RSA-OEAP is a probabilistic encryption scheme that provides semantic security. Semantic security is a property of an encryption scheme that guarantees that a PPT adversary cannot obtain any information about the plaintext from the ciphertext, even if the attacker has access to multiple ciphertexts.

A semantically secure public-key encryption scheme can be considered equivalent to a cryptographic scheme exhibiting the property of Indistinguishability under Chosen-Plaintext Attacks (IND-CPA). An encryption scheme is said to be IND-CPA secure if a PPT adversary, who has access to an encryption oracle $\mathcal{O}^{\mathsf{Enc}}$, cannot distinguish between the encryption of two different plaintext messages, even if they choose these plaintext messages [99].

The RSA-OEAP algorithm is an extensively prevalent public-key encryption scheme that offers robust security guarantees and is widely adopted in practical applications. This algorithm is covered by two widely accepted standards:

- IETF – RFC 8017 'PKCS 1: RSA Cryptography Specifications Version 2.2' section 7.1 [132]
- NIST – SP-800-56B 'Recommendation for Pair-Wise Key Establishment Using Integer Factorization Cryptography' Section 7.2.2 [31].

Next we discuss some of the basic number theory of the RSA-OAEP algorithm to provide context for the correctness of our Part 2 constructions. Our discussions will focus on how the algorithm works in $\mathbb{Z}_N^*$.

Properties of the Ring of Integers Modulo N

The ring of integers modulo N, denoted in $\mathbb{Z}_N^*$, is a fundamental concept in abstract algebra, and it plays a crucial role in the RSA-OAEP algorithm.

Working in $\mathbb{Z}_N^*$ provides a mathematical framework for performing arithmetic operations on numbers coprime to N. In RSA-OAEP the modulus N is a large composite number that is the product of two large prime numbers. As stated in Section 4.3.3, for our constructions we limit the algebraic structure of N to be a Blum integer. The security of RSA-OAEP is based on the fact that it is difficult to factorise N into its constituent prime factors.

Working in $\mathbb{Z}_N^*$ allows the RSA-OAEP algorithm to perform encryption and decryption operations using modular exponentiation using the Square and Multiply algorithm noted in Algorithm 1. Furthermore, because $\mathbb{Z}_N^*$ is a finite ring it ensures that the encryption and decryption operations are well-defined and invertible [76]. In the context of our Part 2 constructions the most relevant property of encryption and decryption working in $\mathbb{Z}_N^*$ is the concept of a unit.

In the ring of integers modulo N every integer which is coprime to N is a unit. More formally, each coprime integer can be stated as $\mathbb{Z}_N^* = \{x \mid x \in (0, N) \land \mathsf{gcd}(x, N) = 1\}$. Furthermore, every unit has a unique multiplicative inverse. That is, for any unit e in the ring, there exists a unique element d such that $ed \equiv 1 \bmod \phi(N)$, where 1 is the multiplicative identity in the ring and where $\phi(N)$ is the Euler totient Function. The Euler totient function describes the cardinality or group order of the ring of integers modulo N and can be stated as $|\mathbb{Z}_N^*| = \phi(N) = (p-1)(q-1)$.

In the RSA-OAEP PKE scheme the public key is set to the tuple (N, e), where N is the modulus which is the product of two large distinct prime numbers, and where $e \in \mathbb{Z}_N^*$. The private key is the tuple (N, d), where d is the decryption exponent such that $d \in \mathbb{Z}_N^*$ and $ed \equiv 1 \bmod \phi(N)$. The decryption exponent d is calculated in polynomial time using EEA.

In Chapter 6, we present a comprehensive description of the RSA-OAEP PKE scheme. However, in the remainder of this section, our emphasis shifts to the correctness of the textbook RSA scheme, which serves as the foundation for RSA-OEAP. Through this examination, we can illustrate the essential connection between the correctness of textbook RSA and the uniqueness of the decryption exponent d, which is tied to its multiplicative inverse e, acting as the encryption exponent.

RSA-OAEP Encryption and Decryption Exponents

In this section we describe the correctness of the selection of our encryption and decryption exponents in our Part 2 constructions which rely on the RSA-OAEP scheme.

Correctness of a PKE scheme. In this section we provide the formal definition of what a PKE scheme is and then describe requirements for a PKE scheme to be correct.

We start with the formal definition of a PKE scheme.

Definition 8. Public-Key Encryption Scheme. A Public-Key Encryption scheme is a cryptographic system which consists of three algorithms: Gen, Enc, and Dec [99].

- The key-generation algorithm Gen takes as input security parameter λ and outputs a pair of keys (pk,sk), where pk is the public key, and sk is the private key.
- The encryption algorithm Enc takes as input a message m from a valid message space $\mathcal{M}$ and the public key pk and outputs a ciphertext c.
- The decryption algorithm Dec takes as input the private key sk and the ciphertext c and outputs a message m or an error $\perp$.

Next we describe what a PKE scheme requires to be considered correct. Briefly stated, for a PKE scheme to be correct it must be the case that when a message m is encrypted using a public key pk and then corresponding ciphertext c is decrypted using the corresponding secret key sk, the original message should be recovered without any loss or alteration. The formal definition of the correctness of a PKE scheme is noted in Definition 9.

Definition 9. The correctness of a PKE scheme. For a PKE scheme to be correct the following must be true: For all messages in the message space: $m \in \mathcal{M}$, and all public and private key pairs output by the generation algorithm: $(\mathsf{pk}, \mathsf{sk}) \xleftarrow{\text{R}} \mathsf{Gen}(1^\lambda)$, it must be the case that the decryption algorithm returns the original message that was encrypted by the encryption algorithm with overwhelming probability:
$\mathsf{PR}\big[m \leftarrow \mathsf{Dec}\,(\mathsf{sk}, c \xleftarrow{\text{R}} \mathsf{Enc}\,(\mathsf{pk}, m))\,\big] = 1 - \mathsf{negl}(1^\lambda)$.

Correctness of RSA-OAEP. In the context of our Part 2 constructions The RSA-OAEP algorithm utilises the textbook RSA PKE scheme for encryption and decryption. However, to address the lack of semantic security in textbook RSA due to its deterministic operation there is additional randomness added through the use of padding, random number generation, hashing, and using the XOR Boolean operator. As stated previously, the full details of the RSA-OAEP algorithm will be covered in Chapter 6. However, for these Part 2 preliminaries we will simplify the RSA-OAEP specific elements as the algorithms $\mathsf{OEAP.Encode}$ and $\mathsf{OAEP.Decode}$.

A brief review of the correctness of textbook RSA in the context of how it is used in RSA-OAEP will provide us with the relevant background to discuss the correctness of our TRE-IA and TIDE chapters. So far we have discussed how RSA-OAEP works in the ring of integers modulo N and how each unit in this ring has a unique multiplicative inverse.

We then discussed that the encryption exponent e is a unit within $\mathbb{Z}_N^*$ and therefore inherits the property of having a unique multiplicative inverse decryption exponent d.

We now have the relevant number theory to provide a summary of RSA-OAEP PKE scheme with a focus on the correctness of the textbook RSA encryption and decryption exponents working in $\mathbb{Z}_N^*$:

First $\mathsf{RSA.Gen}$ runs on the security parameter to output a public and private key pair:$(N, e), (N, d) \leftarrow_R \mathsf{RSA.Gen}(1^\lambda)$.

Next a message $m \in \mathcal{M}$ is OEAP encoded to ensure semantic security and then encrypted with textbook RSA, as seen in Algorithm 5.

Algorithm 5: PKE.Enc encrypts message m using OAEP encoding and textbook RSA.

```
   input  : $((N, e), m)$
 1 $m' \leftarrow_R$ OEAP.Encode($m$)
 2 $c \leftarrow m'^e \bmod N$                         // textbook RSA encrypt
   output: $c$
```

Next the message m is recovered from the ciphertext c by decrypting with textbook RSA and then decoding the OEAP, as seen in Algorithm 6.

Algorithm 6: PKE.Dec decrypts ciphertext C using textbook RSA and OAEP decoding.

```
   input  : $((N, d), c)$
 1 $m' \leftarrow c^d \bmod N$                          // textbook RSA decrypt
 2 $m \leftarrow$ OEAP.Decode($m'$)
   output: $m$
```

We now provide the proof of correctness of the textbook RSA scheme using the Fermat-Euler Theorem.

Theorem 10. The textbook RSA PKE scheme is correct.

Proof. Let $(N, e), (N, d) \leftarrow_R \mathsf{RSA.Gen}(1^\lambda)$, where the public key is $\mathsf{pk} := (N, e)$, and the private key is $\mathsf{sk} := (N, d)$. $\mathsf{RSA.Gen}$ will output the λ bit modulus N which is the product of two distinct odd primes p and q. $\mathsf{RSA.Gen}$ will output encryption exponent e such that $e \in \mathbb{Z}_N^*$ and $\mathsf{gcd}(e, \phi(N)) = 1$. $\mathsf{RSA.Gen}$ will output the decryption exponent d such that $d \equiv e^{-1} \bmod \phi(N)$, where d is calculated using EEA.

Then for all $m \in \mathcal{M}$, the ciphertext c is calculated as $c \leftarrow m^e \bmod N$.

First, as d is the multiplicative inverse of e we have $ed \equiv 1 \bmod \phi(N)$. Next, recall by the by the Fermat-Euler Theorem, noted in Theorem 2, that if $\mathsf{gcd}(e, \phi(N)) = 1$, then $e^{\phi(N)} \equiv 1 \bmod N$. Then, the message m is correctly recovered as follows:
$c^d = (m^e)^d = m^{ed} = m^{1+\alpha\phi(N)} = (m^1)(m^{\phi(N)})^\alpha = (m)(1)^\alpha = (m)(1) \equiv m \bmod N$. □

We now provide an overview of our Part 2 schemes and their use of the RSA-OAEP PKE scheme. Specifically, we will discuss the distinctions in the selection process of the encryption and decryption exponents for each scheme and provide a brief analysis of their correctness. Finally, to ease with the distinction of the TRE-IA and TIDE schemes we will provide a table summarising their key similarities and differences. This table will serve as a convenient reference point while reading the chapters that detail the constructions of the TRE-IA and TIDE schemes.

Encryption and Decryption Exponents in the TRE-IA Scheme. In our TRE-IA scheme, the encryptor sets up the system by first generating the decryption exponent by randomly selecting it in $\mathbb{Z}_N^*$, ensuring that $\mathsf{gcd}(d, \phi(N)) = 1$ is true. Next the encryption exponent is calculated using $e \equiv d^{-1} \bmod N$ using EEA. As e is the multiplicative inverse of d, by the correctness of textbook RSA in Theorem 10, the encryptor generating the parameters has the assurance of the correctness of the TRE-IA scheme which utilises RSA-OAEP.

In the TRE-IA scheme the solver then directly solves an RSW time-lock puzzle to recover d. Although this construction uses the RSA-OEAP scheme for encryption and decryption, the encryptor never releases the encryption exponent e - unlike the standard RSA-OEAP scheme. By maintaining the secrecy of the encryption exponent, this construction provides the additional feature of 'Implicit Authentication' to a standard timed-release encryption scheme. Implicit Authentication will be formally discussed in the Chapter 5.

To solve the RSW time-lock puzzle in the TRE-IA construction, the solver is given the challenge parameters $C \coloneqq (x_0, x_t)$. The solver then recovers the decryption key by sequentially calculating $d \coloneqq \sqrt{x_t} = {x_0}^{2^{t-1}} \bmod N$.

A challenge arises for the solver when recovering the decryption key d because, as stated in Theorem 9, the Chinese Remainder Theorem Isomorphism implies there are four unique solutions to the congruence $x \equiv \sqrt{x_t} \bmod N$. The encryptor knows that d is the unique multiplicative inverse of the encryption exponent e because it uses EEA to calculate e. However, the solver does not have access to $\phi(N)$ and only knows that is has found one of the square roots of $\sqrt{x_t} \bmod N$.

Therefore, in the TRE-IA scheme the solver must be given the confidence that the decryption key that they find is correctly calculated and that it is unique. The correctness of recovering d which is equal to $x_{t-1} \equiv {x_0}^{2^{t-1}} \bmod N$ relies on the correctness of the Square and Multiply Algorithm in Theorem 1. The uniqueness of the decryption d is proven by using the Law of Quadratic Reciprocity in Theorem 5. In Chapter 5 in Theorem 13 we show that because the modulus N is a Blum integer, and because $d \in \mathcal{QR}_N$ we can use Theorem 5 to prove the uniqueness of d to the solver, even though the solver does not know $\phi(N)$.

Encryption and Decryption Exponents in the TIDE Scheme. Similar to the TRE-IA scheme, our TIDE scheme is predicated on the RSW time-lock assumption and it also utilises the RSA-OAEP PKE scheme for message confidentiality. However, TIDE departs from the TRE-IA scheme in two fundamental ways. Firstly, in our TIDE scheme, the encryption exponent is fixed to be $e := 65537$. This is in contrast to the TRE-IA scheme where the encryption and decryption exponent are randomly selected. Secondly, unlike the TRE-IA scheme, the solver does not directly recover the decryption key d, but does so in an indirect manner. The solver in the TIDE scheme recovers the decryption exponent by solving an RSW time-lock puzzle to recover all of the square roots of a quadratic residue which allows them to factor N. Then the decryption exponent is found using EEA as $d \equiv e^{-1} \bmod \phi(N)$.

In the RSA-OAEP scheme the NIST SP-800-56B standard states that the encryption exponent e shall be a number between $65537 \leq e < 2^{256}$. The most common encryption exponent used in practice is the number $2^{16} - 1 = 65537$ [64]. This number is used because it is prime, therefore, ensuring that $\mathsf{gcd}(65537, N) = 1$ will be true. This encryption exponent is also widely adopted because it has a low Hamming weight.

The Hamming weight of a number refers to the total number of non-zero or '1' bits in its binary representation. For example, the Hamming weight of the 65537 is two, since there are two '1' bits in the binary representation of the number '10000000000000001'. Each encryption operation in the RSA-OAEP algorithm will use the Square and Multiply algorithm in Algorithm 1. Therefore, for every 1 in the binary representation of the encryption exponent e both a 'square' operation on line 4 will need to be executed, as well as a 'multiply' operation on line 6. However, for every 0 in the binary representation of the encryption exponent, only a 'square' operation will need to be executed. For this reason having an encryption exponent with a low Hamming weight is ideal because less computation is required for the encryption operation.

Returning to the correctness of the TIDE construction we next observe that the RSW time-lock puzzle given to the solver are terms (x, x_0, x_{-t}). The decryption exponent is recovered by the solver by first calculating $x' \equiv \sqrt{x_0} := (x_{-t})^{2^{t-1}} \bmod N$. Therefore, as noted in Definition 3 because x' is a term on a BBS-CSPRNG it must be a quadratic residue of the modulus N. That is $x' \in \mathcal{QR}_N$.

As part of the RSW time-lock puzzle the solver is also given $x \in \mathcal{QNR}_N^{-1}$, where $x^2 \equiv x_0 \bmod N$. Therefore, once the solver has recovered x' from the RSW time-lock puzzle, they now have $x \in \mathcal{QNR}_N^{-1}$ and $x' \in \mathcal{QR}_N$. Therefore, x and x' must be distinct and are both square roots of $x_0 \bmod N$. As such, having x and x' allows the solver to use Fermat's factorization method, noted in Definition 6, and the weaker congruence of squares condition, noted in Definition 7, to factor the modulus N. Once the factors p and q of the modulus N are recovered by solver, they can calculate the decryption key in polynomial time using EEA to recover $d \equiv e^{-1} \bmod \phi(N) = (p-1)(q-1)$.

In the TIDE construction, the solver recovers d using EEA. Therefore, by the property of working in the ring of integers modulo N, and knowing that $e = 65537$ is prime, the solver knows that d is the unique multiplicative inverse of e. In this way, the solver in TIDE already has a guarantee of the uniqueness of the decryption key d and the correctness of the RSA-OEAP PKE scheme follows more readily than the TRE-IA scheme. The full proof of correctness of the TIDE scheme will be presented in Chapter 6.

4.5 Summary of Part 2 Preliminaries

In conclusion, this chapter has provided a foundational understanding of the core concepts essential for the subsequent chapters detailing the construction of the TRE-IA and TIDE schemes. The discussion on the correctness of the RSA-OAEP PKE scheme highlights the importance of ensuring secure and reliable encryption and decryption processes. Furthermore, we highlighted the common elements shared by both TRE-IA and TIDE, such as their reliance on the RSA-OAEP scheme, the use of RSW time-lock puzzles, and the recovery of decryption keys.

As we progress into the following chapters, we will explore the details of TRE-IA and TIDE, demonstrating how they leverage these foundational concepts to achieve their respective objectives. The TRE-IA scheme extends traditional timed-release encryption by incorporating implicit authentication, enhancing security. Furthermore, TIDE introduces novel approaches to practical delay-based cryptography for use in auctions through the unique use of the recovery of the RSA-OAEP decryption exponents. These innovations represent our research contributions to the field of delay-based cryptography and will be examined and elaborated upon in the subsequent chapters.

Chapter 5

Applications of Timed-release Encryption with Implicit Authentication

A whistleblower is a person who leaks sensitive information on a prominent individual or organisation engaging in an unlawful or immoral activity. Whistleblowing can mitigate corruption and fraud by identifying the misuse of capital. In extreme cases whistleblowing can also raise awareness about unethical practices to individuals by highlighting dangerous working conditions. Obtaining and sharing the sensitive information associated with whistleblowing can carry great risk to the individual or party revealing the data. In this chapter we extend the notion of timed-release encryption to include a new security property which we term implicit authentication*, with the goal of making the practice of whistleblowing safer.*

We formally define the new primitive of timed-release encryption with implicit authentication (TRE-IA), providing rigorous game-base definitions. We then build a practical TRE-IA construction that satisfies the security requirements of this primitive, using repeated squaring in an RSA group, and the RSA-OAEP encryption scheme. We formally prove our construction secure and provide a performance analysis of our implementation in Python along with recommendations for practical deployment and integration with an existing whistleblowing tool SecureDrop.

This chapter appears as part of the Proceedings of AFRICACRYPT 2023, the 14th International Conference on Cryptology in Africa, July 2023, on Pages 490-515, with the title 'Applications of timed-release encryption with implicit authentication' [113]. The research and co-authorship of this chapter included collaboration with Liam Medley and Christian O'Connell. All work was completed under the supervision of Elizabeth Anne Quaglia.

To eliminate redundancy, we have revised Section 5.3 and Section 5.4 in this chapter, deviating marginally from the original AfricaCrypt 2023 publication. This adjustment was made to remove duplication as some of the content was introduced in the preceding Part 2 bridging chapter.

5.1 Introduction

In 2013, Edward Snowden leaked highly classified information from the National Security Agency [153, 164]. This information was leaked at great personal risk. Other recent cases of whistleblowing include the Panama papers [137], the Paradise papers [39], and the Pandora papers [109]. Leaking information subjected the whistleblowers to personal danger due to the power and influence of the organisations whose data was leaked. In the case of the Panama papers, the whistleblower claimed their 'life was in danger' [83].

In this work we construct a cryptographic tool based on timed-release encryption [62], which can augment existing tools for whistleblowers, such as SecureDrop [17]. Our goal is to provide an element of guaranteed delay in the release of sensitive information which has potential to make the practice of whistleblowing safer. We model our solution on the Edward Snowden case, in which all classified material was destroyed before arriving in Russia, in order 'To protect himself from Russian leverage' [1].

We propose a construction which offers the concept of delay-based encryption for whistleblowers, to allow them to rely on cryptographic assurances rather than the trust of a journalist or ombudsman. The technique we introduce allows sensitive information to be encrypted in such a way that a) there is a predictable delay between the *receipt of the ciphertext* encapsulating the leaked information and the *release* of the information, and b) there is no way an adversary can forge a chosen document to insert alongside the genuine documents. This delay will afford the whistleblower time to destroy all classified material after encapsulating the material, and hence ensure their safety. In the Snowden case, the delay would have allowed passage to a safe harbour country without the sensitive information being decrypted until a specified time.

The core idea of our approach is to have two separate keys, an encryption key and decryption key, the latter being encoded as the solution to the *challenge*. The delay starts once the challenge is distributed. The whistleblower keeps the encryption key, used to encrypt the leaked information, and encapsulates its corresponding decryption key with a time-delay, such that it takes at least t time to recover. We provide the whistleblower with the ability to encrypt and distribute ciphertexts under the encryption key, without 'starting the clock' on the time-delay. At a time of their choosing, the whistleblower can distribute the challenge, upon which a sequential computation taking time t will output the decryption key for the ciphertexts. Due to the asymmetric nature of the encryption key and decryption key, once the decryption key is recovered, the whistleblower will still hold the exclusive ability to encrypt more data at a later date.

We formalise this through the introduction of a security property which we term *implicit authentication*, in order to provide the journalist receiving the leaked information with assurance that an adversarial party cannot encrypt a document of their choosing under the encryption key.

Chapter structure and contributions. In the remainder of this section we provide an overview of our construction and discuss relevant related work. We then provide the following topics in subsequent sections in this chapter: In Section 5.2 we formally define the primitive *TRE with implicit authentication* (TRE-IA), giving game-based definitions of the required security properties. In Section 5.3 we present our construction for a TRE-IA scheme, which is based upon the BBS-random number generator and RSA-OAEP encryption. In Section 5.4 we prove our construction is secure under the definitions given in Section 5.2. In Section 5.5 we provide a Python implementation of our construction, along with a performance analysis to demonstrate its practicality. We also provide a practical example of how to use TRE-IA with SecureDrop [17] to show how our construction would integrate with existing whistleblower tools.

5.1.1 Technical Overview

The goal of this work is to explore how a time-delay can be used by vulnerable parties such as whistleblowers in order to make the distribution of sensitive material safer. In order to do so, we introduce a novel construction based upon a clear set of properties, which we can implement in practise and which may augment existing whistleblower tools. Therefore, we define the following security goals that we believe may be helpful to a whistleblower based on the real-life cases of the Panama papers leak [83] and the

Edward Snowden leak [153, 164].

1. An *adjustable time delay*: this will allow the whistleblower to destroy all materials that can be used against them. It also allows the whistleblower to have a configurable amount of time to reach a place of safety.

2. *Maximum flexibility*: this will allow the whistleblower to determine when a) they can encrypt and distribute messages and b) they can 'start the clock' for evaluating the delay. This is achieved through the separation of the ciphertexts and the *challenge*. When the challenge is distributed this starts the clock.

3. *Implicit authentication*: this ensures that no other entity can generate a document of their choice to insert into the leak.

Property 1 can be useful for a whistleblower to protect themselves from the dangers associated with carrying sensitive material. Property 2 allows a whistleblower to gather various different pieces of evidence over time and encrypt and distribute this evidence to journalists. The whistleblower can also ensure that the journalists cannot leak the material until a time delay has passed, thus mitigating risks to the personal safety of the whistleblower. We believe it is crucial that the whistleblower remains in control of all aspects of the system, and by giving the whistleblower the freedom to distribute ciphertexts *without* 'starting the clock' on the time delay, we minimise the trust placed in journalists, and provide the whistleblower with fine-grained control of when the documents are leaked.

Property 3 ensures that once the decryption key has been derived, it cannot be used by third parties to obtain the encryption key to encrypt their own messages. Without this property, it is possible for a third party to choose and encrypt their own fake material, and claim it is from the whistleblower.

We now describe the methodology of how we designed our construction.

Building our construction. Our base property, 1, can be achieved using various primitives, most notably time-lock puzzles (TLPs) and timed-release encryption (TRE). We will start our discussion with TLPs.

A time-lock puzzle [151] encrypts a message to the future, in such a way that once a solver spends a predictable amount of time evaluating the encrypted message, they obtain the plaintext message. One could think of using the naive approach of simply encrypting each message as a TLP and passing it to a journalist. This achieves property

1, however it limits the whistleblower to the condition that they encrypt all materials at once and allows adversarial parties to impersonate the whistleblower. As we wish for the whistleblower to have finer control of the encryption process we see that the latter method has limitations. For example, the whistleblower may wish for multiple messages to be encrypted. This is a reasonable assumption as the Panama papers exposed over eleven million leaked documents [137] and the Paradise papers exposed over thirteen million leaked documents [39]. Using only a basic TLP results in the loss of property 2, that is, the whistleblower loses the element of maximum flexibility. A more appropriate approach is to use a symmetric key as the solution to a TLP, and then distribute ciphertexts separately. This gives us a solution which is close to ideal, as ciphertexts can be distributed to journalists, with guarantees both time delay and flexibility as to when the whistleblower starts the clock. In other words, we can achieve properties 1 and 2 using this approach, however we are missing the implicit authentication property described in 3.

Our approach to fixing this problem is to require that the encryption key and decryption key are different, and more importantly, that one *cannot derive the encryption key from the decryption key.* If this is the case, a whistleblower can encapsulate the decryption key of a public-key encryption scheme, whilst keeping the encryption key secret.

As a starting point, we use TRE as defined by Chvojka et al., who show that one can generically construct TRE from a TLP and a public encryption scheme [62]. However, Chvojka et al.'s approach to constructing TRE does not give us the desired property of a secret encryption key, nor does it necessarily imply that it is impossible to derive the encryption key from the decryption key.

Therefore, to achieve these goals simultaneously we propose a variant of timed-release encryption, which we term TRE with implicit authentication (TRE-IA) and instantiate this with a construction based upon the BBS-CSPRNG random number generator [44], and the RSA-OAEP encryption scheme [37].

5.1.2 Related Work

Time, delay, and encryption There are several primitives in the literature that have considered the component of time and delay in the context of encryption. Time-lock puzzles (TLPs) and the associated timed-release crypto schemes, first proposed by Rivest et al. in [151] are discussed in Section 4.2. Timed-release crypto is closely related to TRE-

IA in that it achieves a very similar functionality. However, there are differences between the two primitives: In essence, TRE-IA uses distinct (asymmetric) keys for encryption and decryption, as opposed to one symmetric key. In both primitives decryption keys are recovered after the delay, in order to decrypt the ciphertext. However, in TRE-IA only the decryption key is recovered. This leads to a different functionality: the whistleblower has control over what information is leaked (i.e., encrypted) as the sole holder of the encryption key.

Previous literature also discusses timed-release encryption (TRE) in two main forms. The classical definition of a TRE scheme is to use a third-party time-server to release messages after a given amount of time [121, 57]. The most recent paper by Chvojka et al. defines TRE as a combination of a time-lock puzzle and public-key encryption [62]. This is in order to introduce the notion of a *sequential* TRE scheme, with the goal of building a public sequential-squaring service which acts as the time server. This allows many parties to time-lock messages, whilst only one party performs the lengthy computation. Conceptually, the classical definition of timed-release encryption differs from the Chvojka et al. scheme because the former relies on a trusted time-server. For the remainder of this work, when we write TRE we will be referring to the method of Chvojka et al.

The main difference between TRE and TRE-IA is the former requires encryption to be public rather than only a prerogative of the whistleblower: In TRE-IA the encryption key cannot be derived by the solver even when the decryption key is recovered. In a TRE-IA scheme assurance is provided through the property of implicit authentication that only the whistleblower can encrypt to the classic notion of a *public key* in a standard PKE scheme.

In the delay-based cryptography literature there are also verifiable delay functions (VDFs), first proposed by Boneh et al. [48] and a new, closely related primitive called Delay Encryption (DE) [53], derived from VDFs. VDFs require a prover to compute a slow sequential computation of length t, also known as a challenge, and then efficiently prove to a verifier that they have done so. VDFs do not generate a ciphertext, or indeed have any method for encrypting data, and so are clearly distinct from TRE.

In DE, the concept of a *session* is introduced. A session consists of a session ID, which any party may encrypt a message to, and an extraction key. Any party may extract the session key, which allows for decryption of all messages encrypted to the session ID, and this extraction takes t time to run. This primitive is distinct from TRE-IA, in that DE allows anyone to encrypt to the session ID, compared to just the whistleblower being in control of encryption in TRE-IA.

Signcryption A cryptographic primitive offering a similar property to implicit authentication (IA) is Signcryption [169, 170, 171]. IA provides the receiving party with assurance that the encrypted documents were sent by the whistleblower. The IA property states that only the holder of the private encryption key can generate a legitimate ciphertext that can be correctly decrypted to a chosen message. In a similar fashion the concept of Signcryption was introduced to provide a single computation that would simultaneously provide the authenticity from a digital signature scheme (DSS), and the confidentiality from a public-key encryption (PKE) scheme. However, Signcryption does not consider the property of delay which is crucial to our TRE-IA scheme.

Tools for whistleblowers A variety of tools exist which can provide protection to whistleblowers. The Tor [16] browser can be used to navigate the Internet anonymously, PGP [172] keys and encrypted email services can support the secure communication between a whistleblower and the investigative journalist, as well as end-to-end encrypted messaging services such as Signal [18]. Closely aligned to our intended end-goal is the SecureDrop [17] submission system, which enables whistleblowers to securely deliver documents containing leaked information. The novelty of our proposed solution in this space is the introduction of a delay, which provides, when combined with encryption, a time 'bubble' within which the whistleblower can reach safety. Indeed, we see TRE-IA as an addition to a whistleblower's toolbox featuring the property of a time-lock delay. In this context, we recognise that no tool is perfect and solutions which guarantee the safety of the whistleblower should be grounded in reality. Our proposed scheme represents a first technological step towards introducing delay as a form of protection to whistleblowers. Accordingly, to provide an example of integrating with existing whistleblowing tooling, in Section 5.5.2 we show how the TRE-IA construction can be augmented to work with the SecureDrop tool.

We conclude this section by noting that the whistleblower use case is an illustrative example of how and why TRE-IA could be useful. We envisage TRE-IA being useful in further applications where a delay and the ability to control the release of sensitive information could help users in at risk situations.

5.2 Timed-Release Encryption with Implicit Authentication

We now provide the definition and properties of TRE-IA, which can be seen as an extension of the TRE primitive as defined by Chovjka et al. [62]. We deviate from the security model of Chvojka et al. in the following way: (i) to fit in with our model of the encryptor alone knowing the encryption key, we require that the encryptor runs setup, (ii) we introduce the new notion of implicit authentication. We provide an intuition of this security property, along with a detailed description of the game in the following section.

In the definition and security games that follow, $\mathcal{E}$ is the encryptor, $\mathcal{D}$ is the decryptor, $\mathcal{A}$ is the PPT adversary, $\leftarrow_{\text{R}}$ represents a probabilistic algorithm, and $\leftarrow$ represents a deterministic algorithm.

Definition of TRE-IA. A TRE-IA scheme consists of the following algorithms: $(\mathsf{TRE.Setup}, \mathsf{TRE.Gen}, \mathsf{TRE.Enc}, \mathsf{TRE.Solve}, \mathsf{TRE.Dec})$, defined as follows.

- $pp, td \leftarrow_{\text{R}} \mathsf{TRE.Setup}(1^\lambda)$. $\mathsf{TRE.Setup}$ is an algorithm run by $\mathcal{E}$ that takes as input security parameter 1^λ and outputs the public parameter pp and trapdoor td. $\mathcal{E}$ must keep td private. The parameter pp can be given to $\mathcal{D}$ after $\mathsf{TRE.Setup}$ completes. $\mathsf{TRE.Setup}$ runs in time $\mathsf{poly}(\lambda)$.

- $e, d, C, t \leftarrow_{\text{R}} \mathsf{TRE.Gen}(pp, td, t)$. $\mathsf{TRE.Gen}$ is an algorithm run by $\mathcal{E}$ that takes as input the public parameter pp, the trapdoor td, and time parameter t and computes an encryption key e, decryption key d, and a public challenge C. The term t indicates the number of sequential steps required to evaluate C to recover the decryption key d when the trapdoor td is not known. The parameter t can be given to $\mathcal{D}$ after $\mathsf{TRE.Gen}$ completes. However, care must be taken when the challenge C is given to the decryptor $\mathcal{D}$. $\mathcal{E}$ must only provide challenge C when they wish to 'start the clock' on recovering the decryption key d. $\mathsf{TRE.Gen}$ runs in time $\mathsf{poly}(\lambda)$.

- $c \leftarrow_{\text{R}} \mathsf{TRE.Enc}(m, e, pp)$. $\mathsf{TRE.Enc}$ is an algorithm run by $\mathcal{E}$ that takes as input a message m, encryption key e, and public parameter pp and outputs ciphertext c. The ciphertext c can be given to $\mathcal{D}$ after $\mathsf{TRE.Enc}$ completes. $\mathsf{TRE.Enc}$ runs in time $\mathsf{poly}(\lambda)$.

- $d \leftarrow \mathsf{TRE.Solve}(pp, C, t)$. $\mathsf{TRE.Solve}$ is an algorithm run by $\mathcal{D}$ that takes as input the parameters pp, C, t and outputs the decryption key d. $\mathsf{TRE.Solve}$ requires t sequential steps to recover d with a run time of $(t)\mathsf{poly}(\lambda)$.

- $\{m', \perp\} \leftarrow \mathsf{TRE.Dec}(c, d, pp)$. $\mathsf{TRE.Dec}$ is an algorithm run by $\mathcal{D}$ that takes as input the ciphertext c, decryption key d, and public parameter pp and outputs plaintext m' or error $\perp$. $\mathsf{TRE.Dec}$ runs in time $\mathsf{poly}(\lambda)$.

A TRE-IA scheme must satisfy the properties of *correctness*, *security*, and *implicit authentication.*

Correctness Intuitively, a TRE-IA scheme is *correct* if any encrypted message can be decrypted successfully to the original plaintext using the corresponding decryption key with overwhelming probability. Namely, in the context of TRE-IA, the legitimate decryption key when input into $\mathsf{TRE.Dec}$ will recover the original message input into $\mathsf{TRE.Enc}$. This is made precise in the Correctness game.

TRE-IA Correctness Game

```
1 $\mathcal{E}$ outputs the public parameter and trapdoor: $pp, td \leftarrow_{\mathrm{R}} \mathsf{TRE.Setup}(1^\lambda)$.
2 $\mathcal{E}$ outputs an encryption key, decryption key, challenge, and time parameter:
  $e, d, C, t \leftarrow_{\mathrm{R}} \mathsf{TRE.Gen}(pp, td, t)$.
3 $\mathcal{E}$ computes the ciphertext on message $m$: $c \leftarrow_{\mathrm{R}} \mathsf{TRE.Enc}(m, e, pp)$.
4 $\mathcal{D}$ recovers the decryption key: $d \leftarrow \mathsf{TRE.Solve}(pp, C, t)$.
5 $\mathcal{D}$ decrypts the ciphertext: $m' \leftarrow \mathsf{TRE.Dec}(c, d, pp)$.
  A TRE-IA scheme is correct if $m = m'$ with probability $1 - \mathsf{negl}(\lambda)$.
```

Security In TRE-IA, similarly to the TRE definition presented by Chvokja et al. [62], *security* is defined as an indistinguishability game as follows:

Let an adversary $\mathcal{A}$ chooses two messages of the same length m_0 and m_1. $\mathcal{E}$ chooses one of these messages at random, which it encrypts and sends to $\mathcal{A}$. $\mathcal{A}$ then gets polynomial time to preprocess upon this ciphertext before receiving the challenge C. $\mathcal{A}$ must then make a guess *before* t sequential steps are computed. The scheme is *secure* if no PPT adversary $\mathcal{A}$ can gain an advantage in guessing which message was chosen by $\mathcal{E}$. This is made precise in the Security game.

Implicit authentication is the new property we introduce in TRE, which ensures that an adversary is unable to *forge* a ciphertext for a message of their choice, hence providing an implicit guarantee that ciphertexts are authentic. In the context of our

TRE-IA Security Game

```
1 $\mathcal{E}$ outputs the public parameter and trapdoor $pp, td \xleftarrow{R} \mathsf{TRE.Setup}(1^\lambda)$.
2 $\mathcal{E}$ outputs an encryption key, decryption key, challenge, and time parameter:
    $e, d, C, t \xleftarrow{R} \mathsf{TRE.Gen}(pp, td, t)$.
3 $\mathcal{A}$ selects two messages of the same length $(m_0, m_1)$ for $\mathcal{E}$.
4 $\mathcal{E}$ uniformly selects $b \in \{0,1\}$, and encrypts $m_b$ as $c \xleftarrow{R} \mathsf{TRE.Enc}(m_b, e, pp)$.
5 $\mathcal{A}$ runs a preprocessing algorithm $\mathcal{A}_0$ on the public parameter and the
    ciphertext, and stores $\mathsf{st} \leftarrow \mathcal{A}_0(pp, c)$.
6 $\mathcal{E}$ sends $C, t$ to $\mathcal{A}$.
7 $\mathcal{A}$ runs a PPT algorithm $\mathcal{A}_1$ which outputs $b' \leftarrow \mathcal{A}_1(\mathsf{st}, C, t)$, where $\mathcal{A}_1$ must run
    in fewer than $t$ sequential steps.
  A TRE-IA scheme is secure if $b = b'$ with probability $\frac{1}{2} + \mathsf{negl}(\lambda)$.
```

TRE-IA Implicit Authentication Game

```
1 $\mathcal{E}$ outputs a key pair $e, d$, the public parameters $pp$, a trapdoor $td$, and a
    challenge $C$: $pp, td \xleftarrow{R} \mathsf{TRE.Setup}(1^\lambda)$, $e, d, C, t \xleftarrow{R} \mathsf{TRE.Gen}(pp, td, t)$.
2 $\mathcal{E}$ sends the public parameter $pp$ to the adversary $\mathcal{A}$.
3 $\mathcal{A}$ returns a target message $m^*$ to $\mathcal{E}$.
4 $\mathcal{E}$ sends $\mathcal{A}$ the challenge $C$, time parameter $t$, and the decryption key $d$.
5 $\mathcal{A}$ is also given access to the encryption oracle $\mathcal{O}^{\mathsf{Enc}}$, which takes as input a
    message $m' \neq m^*$, and returns $c' \leftarrow \mathsf{TRE.Enc}(m', e, pp)$ if the message is valid,
    and $\perp$ otherwise.
6 $\mathcal{A}$ returns a ciphertext $c$ to $\mathcal{E}$.
7 $\mathcal{A}$ wins the game if $m^* \leftarrow \mathsf{TRE.Dec}(c, d, pp)$.
  A TRE-IA scheme has implicit authentication if $\mathcal{A}$ wins the game with
    probability no greater than $\mathsf{negl}(\lambda)$.
```

application, this property ensures that a malicious party cannot insert a document of their choice into the leak provided by a genuine whistleblower.

The idea of modelling security against a ciphertext forgery is inspired by the notions of plaintext integrity and ciphertext integrity [36, 40] in the symmetric encryption setting. More specifically, plaintext integrity states that it should be infeasible to produce a ciphertext decrypting to any message which the sender has not encrypted, and ciphertext integrity requires that it be infeasible to produce a ciphertext not previously produced by the sender, regardless of whether or not the underlying plaintext is 'new' [36].

However, these existing notions do not directly map to the asymmetric-key setting, and TRE in particular, since the adversary, after time t, has access to the secret decryption key. This represents a challenge because it allows the adversary to select elements from the ciphertext space with non-negligible probability, and decrypt them to obtain a plaintext, and present this as a forgery. Whilst any message obtained this way will be not necessarily be 'meaningful', this approach makes a simple analogue of either ciphertext authenticity or plaintext authenticity difficult.

In the asymmetric setting there is the notion of selective security. Selective security is a way to formally define the security of identity-based encryption. Identity-based encryption (IBE) is a cryptographic scheme where a user's public key is derived from its identity information, such as an email address or username, eliminating the need for pre-shared keys or certificates [157].

In an IBE scheme, the system is setup with certain parameters, and there is a master key kept secret. Next, given a public identity (like an email), a private key is created. Each person has their own private key. A message is encrypted with a public identity to create a ciphertext. The ciphertext can be decrypted with the private key associated with the corresponding identity. The system is correct because if you use the right private key for a given identity, you will always get back the original message.

The game-based definition of selective security in the context of IBE consists of a challenger and adversary [50]. The adversary can ask for private keys of any identities it wants (except the one it is attacking). It can also ask to decrypt messages. The phases of the security game are as follows: Phase 1: The adversary makes queries to get private keys or decrypt messages adaptively (depending on the previous responses). Challenge: The adversary chooses two messages (m_0, m_1) and an identity it wants to challenge. The challenger encrypts one of the messages under that identity and sends it back. Phase 2: The adversary continues to ask for private keys or decrypt messages adaptively. The adversary then tries to guess which message was encrypted in the challenge. It wins if it guesses correctly. The adversary's advantage is a measure of how well the adversary did in guessing compared to random chance. If it has a high advantage, it means the encryption scheme is not secure.

The selective security game in an IBE scheme checks if, even when an adversary has a lot of power (can get private keys of other identities and decrypt messages), it still cannot break the encryption scheme by distinguishing between two encrypted messages. If the adversary cannot do significantly better than guessing randomly, the encryption scheme is considered secure.

The IBE notion of selective security does not apply directly to our implicit authentication property. Selective security works in settings where the public key is based on identity details. The public key in our TRE-IA scheme is not based on identity details and is instead generated as the multiplicative inverse of the private key, where the private key is probabilistically selected as the first uniformly sampled integer that is co-prime to the group order of the modulus N. Furthermore, in an IBE scheme and then by definition in the IBE selective security game the Setup algorithm outputs a master key, and this master key is then used in the KeyGen algorithm to generate a new secret key. No master key exists in the TRE-IA scheme. Also, in the IBE selective security game, the hard problem is inserted into the key generation of the identity of the challenged user, and this concept does not exist in the TRE-IA scheme where the public key is not based on the identity of any user.

To overcome the direct ability to model our implicit authentication security game against the symmetric notions of plaintext integrity and ciphertext integrity and the asymmetric IBE notions of selective security, we took the approach of modelling our implicit authentication game as an encryption analogue of *selective* forgery [108, 125], a property used in digital signature schemes where an adversary first commits to a target message m^* and is later challenged to forge a signature for this target message. The key difference in our implicit authentication game is that the adversary is instead asked to output an encryption of the target message, rather than a digital signature.

At a high-level, the TRE-IA game proceeds as follows. The adversary is first asked to output a message m^* that they wish to encrypt. The adversary is then given the decryption key, and access to an encryption oracle. Finally, the adversary is asked to output a ciphertext c to the challenger. The adversary wins the game if c decrypts to the message m^*. We make this precise in the Implicit Authentication game.

5.3 Construction of a TRE-IA scheme

In this section we provide the implementation details of a concrete TRE scheme with implicit authentication. The TLP element of our TRE-IA is derived from the construction of the Blum Blum Shub CSPRNG [44], and as such we name it the BBS-TRE. Our TRE-IA construction also uses the RSA-OAEP PKE scheme.

Recall that implicit authentication states that without access to the encryption key an adversary should not be able to forge a ciphertext for a message of their choice. When RSA is used in practice it is standard procedure to use $e = 65537$ as the public encryption exponent [32]. This does not allow for implicit authentication, as an adversary can guess this. Using any 'standard' fixed encryption key, or a key from a fixed small set will allow an adversary to guess this key, and hence encrypt a message as a ciphertext with more than negligible probability. As such, we design our construction to choose e at random, to ensure that we obtain the implicit authentication property whilst still conforming to the NIST SP-800-56B standard for random public exponent key pair generation [32], Section 6.3.2. Using the BBS-CSPRNG provides an elegant solution to integrating random keys in a TRE-IA setup, as seen in [113].

Next, we briefly recall the notation required for our BBS-TRE introduced in Section 4.3.1. In the pseudo code of our algorithms $:=$ indicates assignment, $=$ indicates equality, $\neq$ indicates inequality, $()$ indicates a tuple, and // denotes a comment. The function `prime`(j) outputs a random j-bit Gaussian prime. The function $\mathcal{U}(a, b)$ uniformly selects an integer that is between $a, b \in \mathbb{Z}$, where $a < b$ and a, b are inclusive. Also, the symbol $\wedge$ indicates logical conjunction (and).

A notable selection of delay-based cryptographic schemes such as time-lock puzzles, verifiable delay functions, delay encryption, and timed-release encryption rely on the Rivest Shamir Wagner (RSW) time-lock assumption [49, 121, 141, 151, 166]. Like other RSW-based delay-based cryptographic schemes our BBS-TRE also relies on the RSW time-lock assumption. The definition of the RSW time-lock assumption and the Square and Multiply Algorithm can be found in Section 4.3.3 under Definition 1 and Algorithm 1 respectively.

We next summarise our BBS-TRE as follows:

$$\begin{aligned}
pp := (N, k_0, k_1, G, H), td := \phi(N) \quad &\xleftarrow{\text{R}} \quad \mathsf{TRE.Setup}(1^\lambda) \\
e := d^{-1} \bmod \phi(N), d := x_0^{2^{t-1} \bmod \phi(N)} \bmod N, \quad & \\
C := (x_0, x_t), t \quad &\xleftarrow{\text{R}} \quad \mathsf{TRE.Gen}(pp, td, t) \\
c \quad &\xleftarrow{\text{R}} \quad \mathsf{TRE.Enc}(m, e, pp) \\
d := \sqrt{x_t} \quad &\leftarrow \quad \mathsf{TRE.Solve}(pp, C, t) \\
\{m, \perp\} \quad &\leftarrow \quad \mathsf{TRE.Dec}(c, d, pp)
\end{aligned}$$

In our BBS-TRE $\mathsf{TRE.Setup}$ outputs the public parameters N, k_0, k_1, G, H. The first parameter is the RSA modulus N which is a Blum integer. A Blum integer is the product of two Gaussian primes i.e., $N = pq$, where $p \equiv q \equiv 3 \bmod 4$ [44]. The modulus being a Blum integer is key requirement for the correctness of our scheme. The parameters k_0, k_1, G, H are the RSA-OAEP parameters which can be seen in detail in Algorithm 12. $\mathsf{TRE.Setup}$ also outputs the trapdoor $\phi(N) := (p-1)(q-1)$ and keeps this parameter private.

Next, the $\mathsf{TRE.Gen}$ algorithm outputs the encryption and decryption keys, the challenge (x_0, x_t) and the time parameter t. In the challenge (x_0, x_t) the term x_0 is a randomly sampled quadratic residue of N, denoted $x_0 \in \mathcal{QR}_N$. The decryption key d is calculated with the trapdoor using Algorithm 1 with the parameters $(x_0, 2^{t-1} \bmod \phi(N), N)$. If $\mathsf{gcd}(d, \phi(N)) = 1$, then in the challenge (x_0, x_t), the term x_t is set to $d^2 \bmod N$ and the encryption key e is set to $d^{-1} \bmod \phi(N)$. Next the $\mathsf{TRE.Enc}$ algorithm inputs a message m and the encryption key e and the public encryption parameters pp to output the ciphertext c using the RSA-OEAP PKE scheme. This scheme deviates from a traditional RSA-OAEP PKE scheme as the encryption key e remains private to ensure the property of implicit authentication.

Next, the $\mathsf{TRE.Solve}$ algorithm sequentially calculates the decryption key d using Algorithm 1 with the parameters $(x_0, 2^{t-1}, N)$. The RSW time-lock assumption tells us that finding the term d will require $t-1$ sequential modular exponentiations to calculate if the trapdoor $\phi(N)$ is not known. Finally, the $\mathsf{TRE.Dec}$ algorithm takes as input the ciphertext c and the decryption key d and the public parameters pp, and outputs either the message m or an error $\perp$ using the RSA-OEAP PKE scheme. We now provide the full details of our BBS-TRE algorithms.

1) $\mathcal{V}$ runs $pp, td \xleftarrow{\text{R}} \mathsf{Setup}(1^\lambda)$ to generate the public parameter and trapdoor, as seen on Algorithm 10. The function $\mathtt{prime}(j)$ on lines 4 and 5 randomly generates j bit primes $p \equiv q \equiv 3 \bmod 4$. That is, $p \xleftarrow{\text{R}} \mathtt{prime}(j)$. This guarantees that N, which is calculated on line 7, is a Blum integer. Next, the trapdoor is set to $\phi(N) := (p-1)(q-1)$. Next

it runs the function params(1^k) which outputs the parameters for the RSA-OAEP PKE scheme. The parameters k_0, k_1 are integers fixed by RSA-OEAP, and the parameters G, H are cryptographically secure hashing functions. The public parameter is set to $pp := (N, k_0, k_1, G, H)$. The public parameter can be released to $\mathcal{D}$ after TRE.Setup is run, but the trapdoor must remain private. TRE.Setup outputs pp, td.

Algorithm 10: $\mathcal{E}$ runs TRE.Setup on security parameter 1^λ to output public parameter pp and trapdoor td.

```
     input  : $1^\lambda$
   1 $p := 0$
   2 $q := 0$
   3 while $p \neq q$ do
   4     $p := \texttt{prime}(\frac{\lambda}{2})$
   5     $q := \texttt{prime}(\frac{\lambda}{2})$
   6 end
   7 $N := pq$
   8 $\phi(N) := (p-1)(q-1)$
   9 $k_0, k_1, G, H \leftarrow \texttt{params}(1^\lambda)$
     output: $pp := (N, k_0, k_1, G, H), td := \phi(N)$
```

2) $\mathcal{E}$ runs $e, d, C, t \leftarrow_R$ TRE.Gen(pp, td, t) to generate the encryption and decryption keys and the challenge, as seen on Algorithm 11. First, TRE.Gen sets the variable gcd to 0. Next, TRE.Gen enters a while loop to generate an appropriate encryption and decryption exponent e, d for RSA-OAEP. This is done by first uniformly selecting $x \in \mathbb{Z}_N^*$ and computing $x_0 \equiv x^2 \bmod N$. Then TRE.Gen evaluates $d \equiv x_0^{2^{t-1} \bmod \phi(N)} \bmod N$. The decryption key d is calculated using Algorithm 1 with the parameters $(x_0, 2^{t-1} \bmod \phi(N), N)$. Note that $\mathcal{E}$ is able to reduce the exponent $2^{t-1} \bmod \phi(N)$ using the trapdoor. The while loop runs until the decryption key d computed on line 6 is coprime to $\phi(N)$. That is, until gcd $:=$ gcd$(d, \phi(N)) = 1$. Once this is found the while loop exits and the term $x_t := d^2 \bmod N$ is calculated and the encryption key e is calculated using the extended Euclidean Algorithm. In Theorem 11 we prove that the while loop will terminate. Furthermore, we prove that in expectation the number of iterations the while loop will require to generate a challenge such that gcd $:=$ gcd$(d, \phi(N)) = 1$ is $\frac{\pi^2}{3} \approx 3.3$. This finding and the associated proof is a key contribution of our concrete TRE-IA construction. Finally, the challenge C is set to the tuple (x_0, x_t). TRE.Gen then outputs the encryption and decryption keys e, d and the challenge, and time parameter C, t. The challenge that $\mathcal{D}$ must solve to recover the decryption key is: for seed x_0, find d such

that $d \equiv \sqrt{x_t} \bmod N$. The encryption key e must remain private, and C and t must only be released to $\mathcal{D}$ once $\mathcal{E}$ would like the decryption key d to be extracted under the RSW time-lock assumption by using the TRE.Solve algorithm.

Algorithm 11: $\mathcal{E}$ runs TRE.Gen on public parameter, trapdoor, and time parameter pp, td, t to create the encryption and decryption exponents and the challenge e, d, C, t.

```
   input : $pp, \phi(N), t$  // $t \in \mathbb{N}$
 1 gcd := 0
 2 while $gcd \neq 1$ do
 3     $x := \mathcal{U}(2, 2^{\frac{\lambda}{2}})$  // $x \in \mathbb{Z}_N^*$
 4     $x_0 := x^2 \bmod N$
 5     $d := x_0^{2^{t-1} \bmod \phi(N)} \bmod N$
 6     gcd := gcd($d, \phi(N)$)
 7 end
 8 $x_t := d^2 \bmod N$
 9 $e := d^{-1} \bmod \phi(N)$  // EEA
10 $C := (x_0, x_t)$
   output: $e, d, C, t$
```

3) $\mathcal{E}$ runs $c \leftarrow_R$ TRE.Enc(m, e, pp) to output ciphertext c, as seen on Algorithm 12. TRE.Enc is the encryption algorithm of the RSA-OAEP PKE scheme. Each step of the algorithm is described on the comments of each line. The final step is to output the ciphertext c. In a TRE scheme the ciphertext c can be released to the decryptor independently of the challenge and time parameter output by TRE.Gen.

Algorithm 12: $\mathcal{E}$ runs TRE.Enc on (m, e, pp) to output ciphertext c.

```
  input : $m, e, pp$
  // $pp := (N, k_0, k_1, G, H)$
1 $m' := m \mid\mid 0^{k_1}$                 // Zero pad to $n - k_0$ bits
2 $r :=$ rand($k_0$)                        // Random $k_0$ bit number
3 $X := m' \oplus G_{n-k_0}(r)$             // Hash $r$ to length $n - k_0$
4 $Y := r \oplus H_{k_0}(X)$                // Hash $X$ to length $k_0$
5 $m'' := X \mid\mid Y$                      // Create message object
6 $c := m''^e \bmod N$                       // RSA encrypt
  output: $c$
```

4) $\mathcal{D}$ runs $d \leftarrow$ TRE.Solve(pp, C, t) to evaluate the challenge and output the decryption key as seen on Algorithm 13. First TRE.Solve calculates the term $\sqrt{x_t}$ by entering the parameters $(x_0, 2^{t-1}, N)$ into Algorithm 1. By the RSW time-lock assumption it

will take $t-1$ sequential steps to calculate d because the trapdoor is not known by the decryptor $\mathcal{D}$. Next, there is an additional check to ensure if $\sqrt{x_t}^2 \bmod N = x_t$ is true. This is an augmentation to the original RSW time-lock puzzle. The purpose of this check is to give the decryptor assurance that the correct decryption key d has been recovered. That is why the term x_t in included in the challenge, to provide the required information to perform this final assurance check for the decryptor. If the condition is true, then d is set to $\sqrt{x_t}$ and output and the algorithm terminates.

5) $\mathcal{D}$ runs $\{m, \perp\} \leftarrow \mathsf{TRE.Dec}(c, d, pp)$ to output the plaintext message m or $\perp$, as seen on Algorithm 14. $\mathsf{TRE.Dec}$ is the decryption algorithm of the RSA-OEAP PKE scheme. Each step of the algorithm is described on the comments of each line. The final step is to output the message m or $\perp$. By the correctness of the RSA-OAEP PKE scheme, if the parameter d extracted by $\mathsf{TRE.Solve}$ under the RSW time-lock assumption has the property $ed \equiv 1 \bmod \phi(N)$ (line 9 of $\mathsf{TRE.Gen}$), then the message m will be recovered. Else, $\mathsf{TRE.Dec}$ will output $\perp$.

Algorithm 13: $\mathcal{D}$ runs $\mathsf{TRE.Solve}$ to evaluate pp, C, t and output the decryption key d.

```
   input  : $pp, C, t$
   // $pp := (N, k_0, k_1, G, H)$, $C := (x_0, x_t)$
1  $\sqrt{x_t} := x_0^{2^{t-1}} \bmod N$
2  if $\sqrt{x_t}^2 \bmod N = x_t$ then
3  |   $d := \sqrt{x_t}$
4  end
   output: $d$
```

We have presented a concrete construction for a TRE-IA based on a RSW TLP and the RSA-OAEP PKE scheme. We have done this by setting up an RSA modulus which is a Blum integer, generating a TLP challenge and a PKE key-pair, then time-locking the decryption key using the TLP. We then integrated our encryption and decryption exponents (the PKE-style key pair) into the RSA-OAEP scheme for the encryption of a message and the decryption of the ciphertext respectively. In the next sections we review the formal security analysis of our scheme and then give a performance analysis of a real implementation of our scheme.

Algorithm 14: $\mathcal{D}$ runs TRE.Dec on (c, d, pp) to recover message m or output $\perp$.

```
   input  : $c, d, pp$
   // $pp := (N, k_0, k_1, G, H)$
 1 $m'' := c^d \bmod N$
 2 $X := \lfloor m'' 2^{-k_0} \rfloor$                       // Extract $X$
 3 $Y := m'' \bmod 2^{k_0}$                                 // Extract $Y$
 4 $r := Y \oplus H_{k_0}(X)$                               // Recover $r$
 5 $m' := X \oplus G_{n-k_0}(r)$              // Recover padded message
 6 $m := m' 2^{-k_1}$                                   // Remove padding
   output: $\{m, \perp\}$
```

5.4 Security Analysis

In this section we provide the security analysis of our concrete BBS-TRE scheme. First we prove that our BBS-TRE is correct and secure. We then prove that our scheme holds the property of implicit authentication.

We first provide proof of the correctness of our scheme. The outline of our proof will be as follows: first we will prove that TRE.Gen will terminate and generate a suitable RSW time-lock puzzle challenge, second we will prove that TRE.Solve will correctly output the decryption key d, third we will prove that the decryption key d is unique because N is a Blum integer, and finally we will prove that the decryption key will correctly return the original message m when it is used to decrypt the ciphertext c generated with the encryption exponent e.

First we must prove that the while loop in TRE.Gen will terminate and generate a suitable challenge and decryption key.

Theorem 11. The while loop in TRE.Gen will in expectation take $\frac{\pi^2}{3}$ trials to generate a suitable challenge and decryption key d.

Proof. The probability of two randomly selected integers being coprime is $\frac{6}{\pi^2}$ [93], Theorem 33. The Blum integer $N = pq$ generated with TRE.Setup is randomly selected using the Miller Rabin Monte Carlo algorithm [126]. Next, the $\phi(N)$ is calculated as $(p-1)(q-1)$. Therefore, $\phi(N)$ is always even. Each iteration of the while loop in TRE.Gen is a Bernoulli trial. In our Bernoulli trial N and hence $\phi(N)$ are randomly selected and the integer d on line 5 of TRE.Gen is also randomly selected. We model d as a random integer as it is an output of the BBS CSPRNG. In each trial, if $\mathsf{gcd}(d, \phi(N)) = 1$ the outcome is a success, otherwise if $\mathsf{gcd}(d, \phi(N)) \neq 1$ then the outcome is a failure. Therefore, in each trial, the probability of selecting two random integers which are coprime when one integer is even, is $\frac{6}{2\pi^2} = \frac{3}{\pi^2}$.

Finally, the probability distribution of the number of Bernoulli trials required until one success is achieved forms a Geometric distribution $G \sim \mathsf{Geo}(\frac{3}{\pi^2})$. Therefore, in expectation, the number of Bernoulli trials required until a suitable challenge and decryption key d is selected such that $\mathsf{gcd}(d, \phi(N)) = 1$ is $\mathbb{E}(G) = \frac{\pi^2}{3} \approx 3.3$ trials.[1] $\square$

Second we must prove that the $\mathsf{TRE.Solve}$ algorithm correctly calculates the decryption key d. To prove this we must show that the Square and Multiply algorithm correctly calculates any term correctly in a BBS sequence. The definition of the BBS CSPRNG and the BBS CSPRNG algorithm can be found in Section 4.3.3 in Definition 2 and Algorithm 2 respectively.

Theorem 12. Algorithm 1 Square and Multiply correctly calculates the x_i term of the BBS CSPRNG.

Proof. The proof of Theorem 12 can be found in the preliminary material in Section 4.3.3 under the proof of Theorem 1. $\square$

Lemma 1. The $\mathsf{TRE.Solve}$ algorithm in the BBS-TRE correctly outputs the decryption key $d \coloneqq x_{t-1}$.

Proof. Suppose encryptor $\mathcal{E}$ honestly generates a random public parameter, challenge and time parameter $pp \coloneqq (N, k_0, k_1, G, H), (C \coloneqq (x_0, x_t), t)$ and presents these to an honest $\mathcal{D}$. Next, suppose $\mathcal{D}$ selects the legitimate evaluation algorithm $\mathsf{TRE.Solve}$ to evaluate (C, t). The $\mathsf{TRE.Solve}$ algorithm will calculate the decryption key d by entering the following parameters $(x_0, 2^{t-1}, N)$ into the Algorithm 1, which will output $d \coloneqq {x_0}^{2^{t-1}} = x_{t-1} \bmod N$. $\mathsf{TRE.Solve}$ will correctly output the BBS term x_{t-1} with overwhelming probability due to the correctness of Algorithm 1 noted in Theorem 12. Therefore, the $\mathsf{TRE.Solve}$ algorithm will correctly output $d \coloneqq x_{t-1}$. $\square$

Next we must prove that the decryption key $d \coloneqq x_{t-1} = \sqrt{x_t} \bmod N$ output by $\mathsf{TRE.Solve}$ is unique. First we must recall that d by definition of being a term in a BBS CSPRNG sequence is a quadratic residue of the modulus N and provide a brief definition. The definition of quadratic residues can be found in Section 4.3.4 under Definition 3.

[1]Numerical analysis also indicated that over thousands of trials, independent of the size of $\phi(N)$, the average number of iterations the while loop must run until a suitable challenge was found was 3.3.

Therefore, we must prove that the solution d to the follow equation is unique:

$$d \coloneqq \sqrt{x_t} \bmod N \tag{5.1}$$

This challenge arises because the Chinese Remainder Theorem isomorphism, indicates that when $N = pq$, where p, q are distinct odd primes, that Equation 5.1 has four distinct solutions [99]. That is, $\pm a \equiv \pm b \equiv \sqrt{x_t} \bmod N$, where $a \neq b$.

The CRT isomorphism can be found in Section 4.4.1 under Definition 5, and the proof that each solution d in Equation 5.1, has four distinct solutions can be found in Section 4.4.1 under Theorem 9. To continue the proof that the TRE-IA scheme is correct we continue with Theorem 13 which is the natural successor of Theorem 9.

Theorem 13. If $N = pq$ is a Blum integer, then the decryption key d in our BBS-TRE extracted by $\mathsf{TRE.Solve}$ is unique.

Proof. If $N = pq$ is a Blum integer with $p \equiv q \equiv 3 \bmod 4$, then $N \equiv 1 \bmod 4$. By the Chinese Remainder Theorem isomorphism every $r \in \mathcal{QR}_N$ has four distinct square roots $\pm a$ and $\pm b$. As N is a Blum integer, by the law of quadratic reciprocity $\mathcal{J}_N(a) = \mathcal{J}_N(-a)$ and $\mathcal{J}_N(b) = \mathcal{J}_N(-b)$, where $\mathcal{J}_N$ is the Jacobi symbol. It must be the case that $a^2 \equiv b^2 \bmod N$, which implies $(a-b)(a+b) \equiv 0 \bmod N$, which implies $(a-b) \mid N$ and $(a+b) \mid N$. That is, without loss of generality $(a-b) = kp$ and $(a+b) = \ell q$, where $k, \ell \in \mathbb{N}$. Therefore, $\mathcal{J}_p(a) = \mathcal{J}_p(b)$ and $\mathcal{J}_q(a) = \mathcal{J}_q(-b)$. As $p \equiv 3 \bmod 4$, the law of quadratic reciprocity tells us $\mathcal{J}_p(-1) = -1$, we have $\mathcal{J}_q(a)\mathcal{J}_p(-1) = \mathcal{J}_q(-b)\mathcal{J}_p(-1)$. This implies that $\mathcal{J}_N(-a) = \mathcal{J}_N(b)$ or written another way $\mathcal{J}_N(a) \neq \mathcal{J}_N(b)$.

Without loss of generality, eliminate the two roots with $\mathcal{J}_N$ equal to -1, say $\mathcal{J}_N(b) = \mathcal{J}_N(-b) = -1$. This leaves $\mathcal{J}_N(a) = \mathcal{J}_N(-a) = 1$. It is the case that only one of $-a$ or a has $\mathcal{J}_p = \mathcal{J}_q = 1$ as $p \equiv 3 \bmod 4$. Therefore, it is this one that is the only quadratic residue modulo N [44].

Returning to our BBS-TRE, by Lemma 1 the term $d \coloneqq x_{t-1} \bmod N$ is correctly calculated by $\mathsf{TRE.Solve}$ and by definition it is a term in a BBS sequence. Therefore, d is a quadratic residue of the modulus N. Therefore, d is the only one of the four distinct square roots of x_t that is a quadratic residue of N. $\square$

For the final element of the correctness proof of our BBS-TRE construction we must prove that the decryption key d will correctly recover the message in an RSA-OAEP scheme.

Corollary 3. The BBS-TRE is correct.

Proof. By Theorem 11 we know that $\mathsf{TRE.Gen}$ will terminate and output a suitable RSW TLP challenge (C, t). By Theorem 12 and Lemma 1 we know that the decryption key d will be recovered by $\mathsf{TRE.Solve}$. By Theorem 13 we know that the decryption key d against the modulus N is unique. From the Fermat-Euler Theorem [84] we know that the decryption key d calculated on line 5 of $\mathsf{TRE.Gen}$ is the same as the decryption key recovered by $\mathsf{TRE.Solve}$ on line 1. From the correctness of the Extended Euclidean Algorithm [99] we know that e calculated on line 9 of $\mathsf{TRE.Gen}$ is the multiplicative inverse of d. Finally, from the correctness of the RSA-OEAP scheme [37] we know that $\mathsf{TRE.Dec}$ will correctly recover the message m using decryption key d from the ciphertext c output by $\mathsf{TRE.Enc}$ using encryption key e with overwhelming probability. □

Next we prove the security of our scheme. To prove the security of our scheme a set of arguments will need to be addressed. The outline of our proof will be as follows: First we prove that finding a square root $\bmod N$ when N is an RSA modulus is equivalent to factoring N. Second we prove that given the public parameters and the RSW challenge an adversary cannot derive the decryption key d in less than t sequential steps. Finally we prove that if $\mathcal{E}$ selects one of two equal length messages to encrypt using $\mathsf{TRE.Enc}$ and outputs ciphertext c, then the only way an adversary can guess with greater than $\frac{1}{2} + \mathsf{negl}(\lambda)$ probability which message was encrypted is to honestly run $\mathsf{TRE.Solve}$ and recover the decryption key d.

Theorem 14. Let $N = pq$, where p and q are λ bit primes. Then given any $r \in \mathcal{QR}_N$, finding x such that $x^2 \equiv r \bmod N$ is equivalent to factoring N.

Proof. The proof of Theorem 14 can be found in the preliminary material in Section 4.3.5 under the proof of Theorem 6. □

The next part of our proof of the security property is to show that the adversary cannot recover the decryption key in less than t sequential steps. If the adversary can recover d in less than t sequential steps then they can output b' equal to b with probability greater than $\frac{1}{2} + \mathsf{negl}(\lambda)$ by decrypting the ciphertext. Our proof is split into two parts: i) when $\mathcal{A}$ attempts to compute d in less than t sequential steps, and ii) when $\mathcal{A}$ attempts to recover $\sqrt{x_t} \bmod N$ using a method that does not use C, t. Therefore, our next step is to prove that the only way d can be recovered without knowing the factors (or trapdoor) of N is to honestly evaluate the challenge in t sequential steps.

Theorem 15. Our BBS-TRE requires t sequential steps to recover the decryption key d.

Proof. Suppose $\mathcal{E}$ honestly generates a random public parameter pp and generates the encryption key, decryption key, challenge, and time parameter e, d, C, t. Next $\mathcal{A}$ selects two messages of the same length m_0 and m_1 for $\mathcal{E}$ to encrypt. $\mathcal{E}$ uniformly selects $b \in \{0, 1\}$ and encrypts m_b. $\mathcal{A}$ produces a PPT algorithm $\mathcal{A}_0$ which pre-processes pp and c and outputs a state $\mathsf{st} \leftarrow \mathcal{A}_0(pp, c)$. $\mathcal{E}$ sends the challenge and time parameter to $\mathcal{A}$. $\mathcal{A}$ produces a PPT algorithm $\mathcal{A}_1$ to output $b' \leftarrow \mathcal{A}_1(\mathsf{st}, C, t)$.

We start by proving part i), that computing d reduces to the RSW time-lock assumption. First we recall from Lemma 1 that $\mathsf{TRE.Solve}$ correctly outputs the decryption key d in t sequential steps, and we know from Theorem 13 that the decryption key d is unique. Next we note that the pre-processing is carried out before the $\mathcal{A}$ is given the challenge and time parameter C, t. Therefore, the probability that $\mathcal{A}$ can compute d in less than t steps is negligible. Specifically, if $\mathsf{TRE.Solve}$ is honestly run, then d is calculated using Algorithm 1 with input $(x_0, 2^{t-1}, N)$. Therefore, by the RSW time-lock assumption, calculating d with Algorithm 1 requires $t - 1$ sequential steps.

Next, suppose $\mathcal{A}$ selects PPT algorithm $\mathcal{A}_{<t}$ to evaluate d in less than $t-1$ sequential steps. However, using such an algorithm $\mathcal{A}_{<t}$ contradicts the RSW time-lock assumption.

What remains is to show that giving $\mathcal{A}$ the challenge and time parameter $C := (x_0, x_t), t$ does not allow them to take square roots $\bmod N$ faster than sequential squaring. To see this, note that by construction x_0 and x_t are the seed term and t^{th} term in a BBS CSPRNG sequence [44]. Under the Generalised BBS assumption [49], knowledge of these terms does not allow finding $d = \sqrt{x_t}$ faster than sequential squaring unless $x_0^{2^{\lambda(\lambda(N))}}$ mod N is calculated efficiently, where $\lambda(N)$ is the Carmichael function [56]. Finding $x_0^{2^{\lambda(\lambda(N))}}$ efficiently is an open problem given by Theorem 9 of Blum et al. [44, 80, 91].

Next, we prove part ii). Observe that finding $d = \sqrt{x_t} \bmod N$ i.e., taking square roots mod N without challenge and time parameter C, t reduces to the open problem of integer factorisation, Theorem 14. Therefore, as N is an RSA modulus we assume it cannot be factored by any PPT algorithm with more than negligible probability. Therefore, the only way a PPT algorithm could recover the unique decryption key d when the factorisation (or trapdoor) of N is not known is to honestly run $\mathsf{TRE.Solve}$ □

Theorem 15 proves that the adversary cannot recover d in less than t sequential steps to win the Security game. Therefore, to conclude our proof of the security property we must demonstrate that the adversary cannot guess b in less than t sequential steps without knowledge of the decryption key.

Theorem 16. Our BBS-TRE scheme is secure.

Proof. We first assume for a contradiction that a PPT adversary $\mathcal{A}$ can win the Security game with a non-negligible advantage.

Let IND-CPA^{RSA} be the standard IND-CPA game for RSA-OAEP [37]. We now recall the well known result that RSA with OAEP padding is IND-CPA secure under the RSA assumption [81]. We will show that any adversary who can break the Security game can also break the RSA assumption.

When choosing the two messages in the Security game the adversary $\mathcal{A}$ has the same available information as they do in the IND-CPA^{RSA} game. We now analyse the difference between the two games when $\mathcal{A}$ receives the challenge and makes a guess. In the Security game, $\mathcal{A}$ is provided with an additional piece of information, the challenge C, but $\mathcal{A}$ is bounded by computational time t.

If an adversary wins the Security game with a non-negligible advantage without evaluating the challenge using algorithm $\mathcal{A}_0$, then they can also break the IND-CPA^{RSA} game with the same algorithm, as C is not provided to $\mathcal{A}_0$.

Next, given the challenge C, t and the state st the sequentiality property of Theorem 15 proves that if an adversary gains a non-negligible advantage by evaluating the challenge in time less than t with $\mathcal{A}_1$ then they are contradicting the RSW-time lock assumption. Therefore, for the adversary to gain a non-negligible advantage in the Security game, they must break either the RSW time-lock assumption, or the IND-CPA^{RSA} security game, and hence the RSA assumption. The adversary will therefore guess the correct message with probability at most $\frac{1}{2} + \mathsf{negl}(\lambda)$, and hence our TRE-IA scheme is secure according to the Security Game presented in Section 5.2. $\square$

We now prove that our scheme has the property of implicit authentication.

Theorem 17. Our TRE scheme provides the implicit authentication property.

Proof. Suppose that the encryptor runs the TRE.Setup and TRE.Gen algorithms. Let $\mathcal{A}$ receive the RSA modulus N and the OAEP parameters (k_0, k_1, G, H), and let m^* be the target message it outputs.

Now let $\mathcal{A}$ receive the challenge C, the time parameter t, the decryption key d and have access to the encryption oracle $\mathcal{O}^{\mathsf{enc}}$.

In our construction d is chosen at random on lines 3 - 5 of $\mathsf{TRE.Gen}$. As we are working in $\mathbb{Z}_N^*$ there is only one multiplicative inverse of d, which is the encryption key e calculated on line 9 of $\mathsf{TRE.Gen}$. To derive e from d requires knowledge of $\phi(N)$, as

$e := d^{-1} \bmod \phi(N)$. In order to learn $\phi(N)$, the adversary would need to factor the RSA modulus N, which is a well known hard problem [145, 150]. Therefore, unless the adversary can factor N, they cannot guess e with more than negligible probability. Therefore with overwhelming probability the adversary will not learn the trapdoor $\phi(N)$, and hence will not be able to derive e from d.

Using the encryption oracle $\mathcal{O}^{\mathsf{enc}}$ $\mathcal{A}$ can obtain polynomially many ciphertexts.

Recall from [37] that RSA-OAEP has ciphertext indistinguishability under chosen-plaintext attack, which guarantees indistinguishability between encryptions of messages. This property guarantees in particular that the adversary has no advantage in identifying a ciphertext that will allow them to win the IA game.

The adversary can choose random elements from the ciphertext space and decrypt them using the decryption key d. However, without knowledge of the encryption key, any such ciphertext will decrypt to a random element of the message space.

As the size of the ciphertext space is exponential (explicitly it is the magnitude of $\mathbb{Z}_N^*$), and the adversary runs a PPT algorithm, there is a negligible chance of correctly guessing a ciphertext which decrypts to the target message m^*. Therefore the adversary will not win the implicit authentication game with greater than negligible probability. □

5.5 Performance and Integration with SecureDrop

In this section we present an evaluation of how our TRE-IA scheme performs and provide a sample of how it can integrate with SecureDrop. We evaluate the run times of the algorithms executed by the encryptor (who is the whistleblower) and those executed by the decryptor. Our analysis compares the dispersion of run times of TRE-IA algorithms in our construction when different parameters are used for the modulus size. Based on the results of our analysis we present a number of recommendations for TRE implementations.

5.5.1 Performance Analysis

Our test setup consists of a virtual machine running on an Intel i7-6700K 4 GHz CPU and 24 GB of RAM. The guest OS is Ubuntu 18.04 running Python 3.8.10. The `Crypto` library 2.6.1 is used to generate random Blum integers. The code can be found in https://github.com/nxd0main/TRE-IA.

Our performance analysis consists of running six different trials. Each trial tests

Table 5.1: Performance analysis trial parameters and descriptive statistics for TRE scheme run time tests. $\mu^*_{\mathcal{E}}$, $\mu_{\mathcal{E}}$, $\sigma_{\mathcal{E}}$, and $c_{\mathcal{E}}$ are the aggregated encryptor algorithms run time mean without the TRE.Setup algorithm, run time mean with the TRE.Setup algorithm standard deviation, and coefficient of variance, respectively. $\mu_{\mathcal{D}}$, $\sigma_{\mathcal{D}}$, $c_{\mathcal{D}}$, and $\beta_{\mathcal{D}}$ are the aggregated decryptor algorithms run time mean, standard deviation, coefficient of variance, and excess kurtosis, respectively.

Trial	**Modulus [b]**	**runs**	t	ℓ	$\mu^*_{\mathcal{E}}$ [s]	$\mu_{\mathcal{E}}$ [s]	$\sigma_{\mathcal{E}}$ [s]	$\mathbf{c}_{\mathcal{E}}$	$\mu_{\mathcal{D}}$ [s]	$\sigma_{\mathcal{D}}$ [s]	$\mathbf{c}_{\mathcal{D}}$	$\beta_{\mathcal{D}}$
1	2048	200	110^7	50	1.14	1.33	0.13	0.095	121.08	2.01	0.017	-0.18
2	2560	200	110^7	50	2.13	4.74	1.71	0.360	166.37	2.27	0.014	0.12
3	3072	200	110^7	50	3.57	4.07	0.33	0.081	221.41	2.25	0.010	-0.36
4	3584	200	110^7	50	5.54	12.08	4.56	0.378	284.07	2.93	0.010	0.15
5	4096	200	110^7	50	8.13	9.41	0.85	0.090	354.77	3.41	0.010	1.33
6	4608	200	110^7	50	11.31	25.02	8.38	0.335	432.73	4.08	0.009	4.98

how different parameters impact the encryptor and decryptor algorithms run times on our TRE-IA scheme implementation. Trial 1 is tested against a 2048 bit modulus, with the time parameter t set to 10 million. Trial 1 is run 200 times. Each time Trial 1 is executed we record the individual run times of the TRE.Setup, TRE.Gen, TRE.Enc, TRE.Solve, and TRE.Dec algorithms. Trial $2, 3, 4, 5$ and 6 are similar to Trial 1 except they are tested against a $2560, 3072$, 3584, 4096, and 4608 bit modulus, respectively. Table 5.1 summarises the trial parameters and a number of key descriptive statistics based on our analysis.

Encryptor Analysis. We first provide an analysis of the distribution of the aggregated run time of the algorithms run by the encryptor $\mathcal{E}$. For each of the 200 iterations of the six trials in Table 5.1, the aggregated mean run time of the encryptor algorithms is noted under column $\mu_{\mathcal{E}}$ [s]. This column is the sum of the individual run times for TRE.Setup, TRE.Gen, and TRE.Enc algorithms. In the column noted $\mu^*_{\mathcal{E}}$ [s], the aggregated mean run time of the encryptor algorithms without TRE.Setup is recorded to illustrate the overhead of this algorithm.

Table 5.1 also shows the standard deviation $\sigma_{\mathcal{E}}$ [s], and the coefficient of variance $c_{\mathcal{E}}$ for each trial. $c_{\mathcal{E}}$ allows us to normalise the dispersion of the run times around the mean for the different trials [20]. We see that the coefficient of variance is smaller for the $2048, 3072$, and 4096 bit modulus indicating a tighter run time around the mean. Tighter dispersion is ideal to give a more consistent experience. As the encryptor may be under duress, knowledge of run times for operation would be essential.

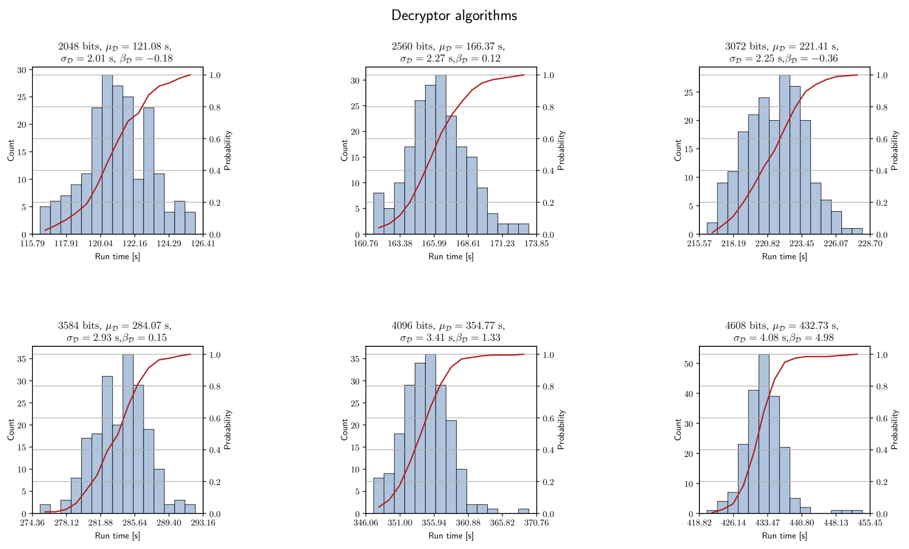

Figure 5.1: Histogram with EDF for our BBS-TRE scheme Decryptor TRE.Solve and TRE.Dec algorithm run times. Heavy tailed distributions with high excess kurtosis can be seen on the 4096 bit and on the 4608 bit trial data. These trials show the probability of outliers is higher than a normal distribution. This can give a decryptor an inconsistent run time when extracting the decryption key and decrypting the ciphertext. If the selection of time parameter t is chosen to provide a specific time for the decryption of data for maximum impact, using parameters with a high probability of outlier run times is undesirable.

Recommendations: Based on the analysis of encryptor algorithm performance, we recommend choosing the 2048 bit and 3072 bit moduli sizes to ensure fast and consistent run times demonstrated by their low mean run times and low coefficients of variance respectively.

Decryptor Analysis. Next, we provide analysis of the distribution of the aggregated run times of the algorithms run by the decryptor $\mathcal{D}$. The aggregated decryptor algorithm run time equals the sum of the TRE.Solve and TRE.Dec algorithms. The mean $\mu_{\mathcal{D}}$, standard deviation $\sigma_{\mathcal{D}}$, coefficient of variation $c_{\mathcal{D}}$, and excess kurtosis $\beta_{\mathcal{D}}$ for each trial can be seen in Table 5.1. In Table 5.1 an upward trend exists for the mean decryptor run times as the size of the modulus increases. This is expected as more bit operations are required for a larger modulus. Increasing the modulus size from 2048 bit to 4608 bit causes the mean decryptor run time to grow from 121.08s to 432.73s. This shows that a 2.25 times increase in the modulus size is proportionate to a 3.6 times increase in the mean decryptor run time. The coefficient of variation remains consistently low between all trials in Table 5.1. This is ideal because having a low coefficient of variance for the decryptor algorithms indicates a tighter dispersion around the mean. By this measure, the decryptor algorithm run times appear to be consistent.

We also measure the dispersion properties of the decryptor run times using the excess kurtosis, denoted $\beta_{\mathcal{D}}$. Excess kurtosis provides information about the weight of the tails of a distribution [167]. Excess kurtosis also provides an indicator of the probability of outliers in the distribution. Therefore, lower excess kurtosis is ideal to provide assurance of consistent run times. A standard normal distribution has an excess kurtosis of 0, known as a mesokurtic distribution. If the excess kurtosis of a distribution is > 0 the distribution is said to be leptokurtic. Leptokurtic distributions have a higher probability of outliers, that is, they are heavy tailed. If the excess kurtosis of a distribution is < 0 the distribution is said to be platykurtic. Platykurtic distributions have a lower probability of outliers, that is, they are sparsely tailed.

Figure 5.1 plots the histograms of the decryptor run times for each of the six trial types in Table 5.1. Each histogram also plots an empirical distribution function (EDF). Figure 5.1 shows us that the 2048 and 3072 bit BBS-TRE distributions show the lowest probability of outlier run times and that the 4096 bit and 4608 bit distributions show the highest probability of outlier run times.

Figure 5.1 shows us that the 2048 and 3072 bit TRE scheme distributions are platykurtic – that is, they demonstrate a shorter tailed distribution on their EDFs with a lower probability of outlier decryptor run times. The figure also shows us that the 2560 bit and 3584 bit TRE scheme distributions are slightly leptokurtic – that is, they demonstrate a slightly longer tailed distribution on their EDFs with a moderate probability of outlier decryptor run times. Finally, Figure 5.1 shows us that the 4096 bit and 4608 bit TRE scheme distributions are strongly leptokurtic – that is, they demonstrate a long heavy tailed distribution on their EDFs with a strong probability of outlier decryptor run times. Further analysis of the decryptor algorithm data shows us that for the platykurtic distributions there were 0 trials that were $> 3\sigma_{\mathcal{D}}$ from the mean. Conversely, the slightly platykurtic distributions each had 1 outlier $> 3\sigma_{\mathcal{D}}$, respectively. Finally, the strongest platykurtic distribution had 4 outliers $> 3\sigma_{\mathcal{D}}$. The empirical rule for normal distributions tells us that only 0.3% of the population should fall $> 3\sigma$ from the mean. However, we see that our TRE scheme 4608 bit construction had 4% of the values $> 3\sigma_{\mathcal{D}}$ from the mean.

The presence of heavy tailed distributions in our trials was unanticipated as the

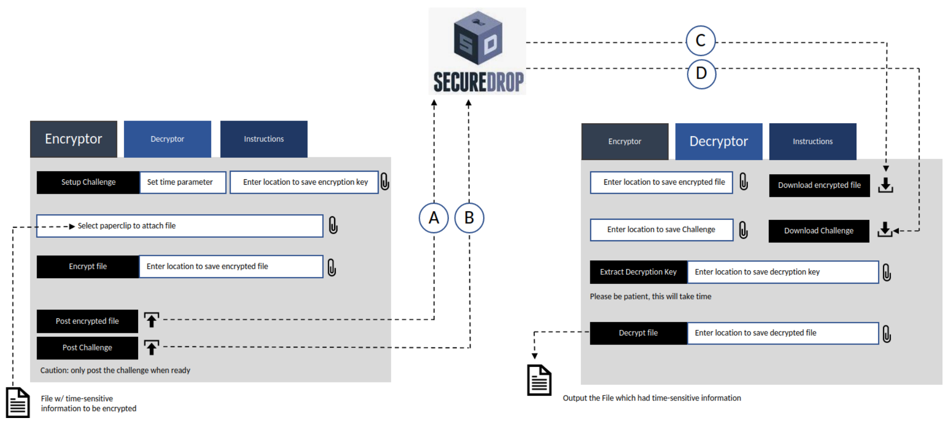
SECUREDROP
C
D
A
B
Encryptor
Decryptor
Instructions
Setup Challenge
Set time parameter
Enter location to save encryption key
Select paperclip to attach file
Encrypt file
Enter location to save encrypted file
Post encrypted file
Post Challenge
Caution: only post the challenge when ready
File w/ time-sensitive information to be encrypted
Encryptor
Decryptor
Instructions
Enter location to save encrypted file
Download encrypted file
Enter location to save Challenge
Download Challenge
Extract Decryption Key
Enter location to save decryption key
Please be patient, this will take time
Decrypt file
Enter location to save decrypted file
Output the File which had time-sensitive information

Figure 5.2: SecureDrop integration with TRE-IA.

virtual machine resources and test parameters remained consistent. Having heavy tailed distributions with a high probability of outliers for decryptor algorithm run times when the same computational resource is provided is undesirable. For example, a decryptor extracting and decrypting the ciphertext on a 4608 bit construction would know from performance analysis that there is a 1 in 25 chance that they could take far longer than the expected mean time to extract the time-locked decryption key and decrypt the ciphertext. By the measure of excess kurtosis, the decryptor algorithms run times appear to vary considerably, with some parameters displaying more ideal outcomes than others.

Consider a concrete example extrapolated from the final histogram in Figure 5.1 for the 4608 bit modulus. If $\mathcal{E}$ selected the parameter t such that the expected extraction and decryption time was 7 days (168 hours), then a decryptor may take 8 hours longer than the expected run time to recover the plaintext.

Recommendations: Based on the analysis of decryptor algorithm performance, we recommend using the 2048 and 3072 bit sizes which demonstrate the lowest probability of outlier run times. First, we recommend using the modulus sizes which demonstrate the lowest probability of outlier run times. The two moduli sizes with this property are the 2048 bit and 3072 bit constructions – which correspond with the best moduli sizes to use by the whistleblower. Second, we recommend that the 4096 and 4608 bit constructions should be used with caution because they exhibit leptokurtic excess kurtosis, thus the probability of outlier run times is higher.

5.5.2 Using TRE-IA with SecureDrop

Our TRE-IA construction could augment an existing whistleblowing tool known as SecureDrop by adding the property of guaranteed delay with implicit authentication. SecureDrop is ‘an open-source whistleblower submission system that can securely access documents from anonymous sources’ [17].

The Scope of Our Research In Figure 5.2 we illustrate a possible integration mechanism of the TRE-IA scheme with SecureDrop and provide a visual representation of the workflow. The figure showcases the different steps and components involved in the time-release encryption process.

At this point in the research, the integration of the TRE-IA scheme with SecureDrop is depicted in the figure to provide an illustrative overview only rather than being fully developed and implemented solution. The focus of this chapter’s research was on the understanding of the theoretical aspects of TRE-IA and to provide an initial view of the feasibility of the integration with SecureDrop.

By providing an illustrative figure, readers may grasp the overall conceptual flow of the integration and allow for any peer review before examining the technical implementation details. Fully implementing the integration would be an extensive task requiring significant development effort, infrastructure setup, and thorough testing.

The testing from whistleblowers themselves would be essential in order to obtain valuable insights into the usability, effectiveness, and real-world implications of the integrated TRE-IA scheme with SecureDrop. Whistleblowers, as the primary users of such a system, can provide crucial feedback based on their first-hand experience and perspective.

Additionally, the social science considerations of whistleblower testing would require the utmost respect and consideration. Given the sensitive nature of whistleblowing and the potential risks involved, extensive social science considerations would be necessary to test the efficacy of such tooling. These considerations encompass understanding the psychological and social dynamics of whistleblowing, the potential impact on the individuals involved, and the ethical aspects of implementing such a system. Social scientists can help design protocols for participant engagement, ensuring their safety, well-being, and informed and ongoing consent throughout the testing process. We will discuss this further in our concluding Chapter regarding the outlook of our research.

Therefore, presenting an illustrative figure allows researchers to convey the integration concept without diverting resources from the primary research objectives of this chapter which was the exploration of TRE with the property of Implicit Authentication.

Illustrating TRE-IA Integration with SecureDrop In the figure, the left-hand box represents the 'Encryptor' tab for the Whistleblower. In this tab is the 'Setup Challenge' phase can be executed where the $\mathsf{TRE.Setup}$ and $\mathsf{TRE.Gen}$ algorithms are executed, allowing the selection of the time parameter. Additionally, there is a dialogue box to securely save the encryption key e to maintain the property of Implicit Authentication. The 'Encrypt file' dialogue box enables the selection of files for time-release encryption using $\mathsf{TRE.Enc}$.

Regarding the whistleblower's perspective, the figure highlights that the whistleblower can upload the public parameters and ciphertext at their preferred time, indicated by the dashed-line denoted as A. The 'Post Challenge' button allows the upload of the challenge to SecureDrop, depicted by the dashed-line B.

On the right-hand side, the figure represents the 'Decryptor' tab for the receiving party. The decryptor can download and save the ciphertext as soon as it becomes available on SecureDrop, shown by dashed-line C. However, they cannot decrypt the ciphertext until the challenge becomes available and is downloaded, indicated by dashed-line D. Once the challenge is received, the 'Extract Decryption Key' button triggers the $\mathsf{TRE.Solve}$ algorithm to recover the decryption key d. Finally, the 'Decrypt file' option utilises $\mathsf{TRE.Dec}$ along with the ciphertext, public parameters, and decryption key to recover the original leaked file.

By referring to Figure 5.2, we can gain a visual understanding of how the TRE-IA scheme integrates with SecureDrop, facilitating secure and time-release encryption.

5.6 Conclusion

In this chapter we introduced a variant of a delay-based primitive known as timed-release encryption with implicit authentication (TRE-IA). Implicit authentication is formally introduced with a game-based definition and we provide a concrete implementation with this property. Our implementation of a TRE-IA which we name the BBS-TRE uses the BBS CSPRNG and RSA-OAEP PKE as the building blocks. Our construction is implemented in Python and a performance evaluation against six common moduli sizes was provided. Our performance analysis allowed us to observe how the modulus size affected the run time consistency of the whistleblower and decryptor algorithms. Therefore, we were able to provide concrete recommendations for parameter selection for practical implementations of our TRE scheme. We also provided an example of how our TRE-IA scheme could be used in conjunction with the existing whistleblowing tool SecureDrop.

Chapter 6

TIDE: A Novel Approach to Constructing Timed-Release Encryption

Chvojka et al. introduced the idea of taking a time-lock puzzle and using its solution to generate the keys of a public-key encryption (PKE) scheme [62]. They use this to define a timed-release encryption (TRE) scheme, in which the secret key is encrypted 'to the future' using a time-lock puzzle, whilst the public key is published. This allows multiple parties to encrypt a message to the public key of the PKE scheme. Then, once a solver has spent a prescribed length of time evaluating the time-lock puzzle, they obtain the secret key and hence can decrypt all of the messages.

In this work we introduce TIDE (TIme Delayed Encryption), a novel approach to constructing timed-release encryption based upon the RSA cryptosystem. In our timed-release encryption scheme instead of directly encrypting the secret key to the future we utilise number-theoretic techniques to allow the solver to factor the RSA modulus which allows the derivation of the decryption key. We implement TIDE on consumer grade hardware including testing on a desktop PC and on Raspberry Pi devices validating that TIDE is both efficient and practically implementable. We provide evidence of practicality with an extensive implementation study detailing the source code and practical performance of TIDE.

This chapter appears as part of the Proceedings of ACISP 2022, the Australasian Conference on Information Security and Privacy, November 2022, on Pages 244-264, with the title 'TIDE: A Novel Approach to Constructing Timed-Release Encryption' [112]. The research and co-authorship of this chapter included collaboration with Liam Medley. All work was completed under the supervision of Elizabeth Anne Quaglia.

To limit duplication, we have revised Section 6.2, Section 6.3, and Section 6.4 in this chapter, with minor edits from the original ACISP 2022 publication. This adjustment was necessary as some of the content was introduced in the Part 2 bridging chapter.

6.1 Introduction

In 1996, Rivest et al. introduced the notion of sending a message 'to the future' using a time-lock puzzle [151] as introduced in Section 4.2. Delay-based cryptography is a prominent and wide-ranging subject built around the notion of associating standard 'wall-clock time' with an iterated sequential computation. In current research delay-based cryptography is used in the classical sense of encrypting a message to the future using various primitives such as time-lock puzzles [120, 79], timed-release encryption[62, 57], and delay encryption [53]. There are also alternative applications of delay-based cryptography such as providing a computational proof-of-age of a document and constructing public randomness beacons [48, 141, 166].

In this chapter, we introduce a novel construction of timed-release encryption named timed delay encryption (TIDE). TIDE is well known suited to the application of sealed-bid auctions. TIDE can provide a practical and efficient solution to Vickrey auctions which we now explore.

6.1.1 Sealed-bid Auctions

Sealed-bid auctions allow bidders to secretly submit a bid for some goods without learning the bids of any other party involved until the end of the auction. In a sealed-bid second-price auction – known as a Vickrey auction – the highest bidder wins the goods but pays the price of the second highest bid [165, 28, 45]. The challenge of building a transparent, fair, efficient, and cryptographically secure Vickrey auction has been of interest for decades [53, 97, 82, 43, 51].

In the context of auctions, *fairness* is discussed in Galal et al. in the following manner: if bidders attempt to deviate from the auction protocol and prematurely abort to affect the behaviour of the auction they are financially penalised whilst honest bidders are refunded [82]. Galal et al. also defines *transparency* as the ability for information to be accessible to everyone to check if there is any manipulation or corruption [82].

A common approach to constructing sealed-bid auctions is to implement a *commit and reveal* solution using an append-only bulletin board, i.e., a blockchain [82].

In the context of cryptography, the commit-and-reveal paradigm is a method used to ensure fairness, honesty, and security in auction scenarios, specifically in sealed-bid auctions [99].

During the initial commitment phase, each bidder submits a commitment, which is essentially a sealed envelope containing their bid. This commitment is generated using a cryptographic one-way function or a hash function. The key characteristic is that once a commitment is made, it is computationally infeasible to determine the actual bid from the commitment alone. This phase ensures that bidders cannot change their bids after seeing the bids of others, preventing bid manipulation and late bidding.

During the reveal phase, after all bidders have submitted their commitments, they proceed to reveal their actual bids. This involves providing the original bid value along with information that proves that the revealed bid corresponds to the earlier commitment. The cryptographic proof used here ensures that no bidder can cheat by revealing a different bid than what they initially committed to.

The commit-and-reveal paradigm adds a layer of security and fairness to auctions, especially in scenarios where participants may be concerned about revealing their bids too early. It prevents a bidder from strategically waiting to see the other bids before deciding on their own, as they must commit to their bid before knowing what others are offering. This enhances the trust and integrity of sealed-bid auctions.

However, the main drawback of this approach is that parties are not obliged to open their bids. This is particularly problematic in Vickrey auctions where it is necessary to learn the second highest bid as well as the first [28, 45]. For an auction to be transparent and fair it is a requirement that each party must open their commitments to the bid once the bidding phase has ended.

By replacing the commitments with time-lock puzzles an elegant solution is created to solve this problem [151]. Each party encrypts their bid as the solution to a time-lock puzzle. Therefore, in the opening phase if a party does not reveal their bid it can instead be opened by computing the solution to the puzzle. However, this method does not scale well because it leads to many different time-lock puzzles being solved, which is computationally expensive. Current research attempts to solve this problem more efficiently.

Malavolta et al. [120] suggest that each party encrypts their bid as a time-lock puzzle (TLP), as in the classical method suggested by Rivest et al. Their insight is that the party tallying the bids then uses techniques from homomorphic encryption to evaluate a computation over the set of puzzles to determine the winning bidder. This leads to only the relevant puzzle being solved rather than the entire set of puzzles. Whilst this is a very elegant solution the application relies on fully homomorphic TLP constructions. Currently all constructions of homomorphic TLPs are based on Indistinguishability Obfuscation (IO) [53, 120]. IO aims to obfuscate programs to make them unintelligible whilst retaining their original functionality [30]. However, IO is known to be impractical with no construction efficiently implementable at the time of writing [96].

Burdges et al. introduce Delay Encryption (DE) [53], a primitive which offers an alternative approach to solving the problem of not revealing a bid. Their solution uses a delay-based analogue to identity-based encryption. Time-lock puzzles require each bidder to encrypt their bid against a unique time-lock puzzle. In contrast, DE requires bidders to encrypt their bid to a public *session ID*. This session ID acts as a bulletin board. That is, anyone who knows this session ID can efficiently encrypt messages to it. All messages encrypted to the session ID can be efficiently decrypted by any party who knows a secret *session key*. The key feature of DE is a slow and sequential Extract algorithm which outputs a session key after a prescribed amount of time. This time delay defines the bidding phase of the auction in which parties may encrypt bids to the session ID. Once the session key has been extracted all bids can be decrypted, thus replacing the opening phase described in the commit-and-reveal paradigm. DE works well in the context of auctions for the opening phase. In DE rather than solving multiple time-lock puzzle the Extract algorithm is run once which outputs a *session key*.

DE appears to be an ideal solution to ensure that bids are revealed. However, the construction presented in [53] comes with two practical disadvantages. Firstly, the storage requirements required to compute the decryption key is huge – a delay of one hour requires 12 TiB of storage. Secondly, the time taken to run the initial Setup algorithm grows proportionally to the delay, which is computationally expensive. These two factors make this construction problematic from a practical standpoint.

The goal of the approaches outlined above is to utilise a time-delay to solve the auction problem in a scalable manner. This improves upon the efficiency of Rivest's solution by ensuring that at most two sequential computation (namely the puzzles containing the two highest bids in [120], and Extract in [53]) needs to be run, rather than one for each bid. However, the approaches so far have practical problems with the instantiation of their proposed candidate.

Chvojka et al. introduce the idea of taking a TLP and using its solution in the key generation of a public-key encryption (PKE) scheme [62]. They use this to define a timed-release encryption (TRE) scheme where multiple parties encrypt a message to the public key of the PKE scheme. Upon solving the puzzle they can reconstruct the secret key and decrypt all of the messages. The authors explain how to achieve this generically using standard TLP and PKE primitives, but no concrete instantiation is provided.

In this work we present TIDE, a novel, efficient, and easily implementable approach to building a TRE scheme to solve the scalability problems in Vickrey auctions. TIDE seamlessly integrates an RSA-OEAP PKE scheme into a TLP using classic number theory. As well as being a concrete construction, TIDE subtly differs in its approach to how the secret key is derived when compared to the techniques used in Chvojka et al.

6.1.2 Technical Overview

TIDE relies on the RSW time-lock assumption which states that it is hard to compute $x^{2^t} \bmod N$ in fewer than t sequential steps for an RSA modulus N. The RSW assumption was discussed in Definition 1. It has been used to build a variety of cryptographic constructions [79, 120, 48, 141, 166]. TIDE deviates from previous literature by using number theoretic techniques to utilise the output $x^{2^t} \bmod N$ in a novel way. Previous approaches used squaring solely for its sequential properties, i.e., the final output is used only to guarantee a delay. For example, in time-lock puzzles, the solution to the puzzle is precomputed using a trapdoor, in order to hide a message as the product of the solution and the message [151, 79, 120]. This allows one to obtain the message upon computing the delay, hence solving the puzzle.

In the context of verifiable delay functions (VDFs), the RSW assumption is used to prove that a certain amount of clock time has taken place.

In TIDE the output of the computation expands beyond guaranteeing a delay. Namely TIDE provides exactly the information required to factor the RSA modulus N. This is achieved by incorporating classic number theoretic results from Fermat and Rabin. These results state that if x and x' are known such that $x^2 \equiv x'^2 \bmod N$, where $x \not\equiv \pm x' \bmod N$, then the non-trivial factors of N can be recovered in polynomial time [145].

By carefully setting up the system we provide the user with value x and ensure that the output of the squaring reveals x'. Therefore knowledge of x and x' can be used to factor N in polynomial time. Then we combine this with a standard RSA-OEAP PKE scheme using the modulus N and an encryption exponent as the public key. Once a solving party computes the delay they can derive the secret key and hence can decrypt all messages.

Therefore, our construction can be seen as a natural integration of an RSW-based time-lock puzzle and the RSA-OEAP PKE scheme. We formalise this in terms of syntax and security definitions in Section 6.4, where we follow the definition of TRE by Chvojka et al. [62].

The key insight of TIDE is contained in the generation of the public key and puzzle, as this allows us to use the relevant theorem of Rabin [145]. N is chosen to be a particular class of RSA modulus known as a Blum integer $N = pq$, which has the property that $p \equiv q \equiv 3 \bmod 4$. The puzzle consists of three different elements, $P = (x, x_0, x_{-t})$. First, the element x is efficiently sampled such that $\mathcal{J}_N(x) = -1$, where $\mathcal{J}_N(x)$ is the Jacobi symbol [99]. Next, the seed x_0 is calculated as $x_0 \equiv x^2 \bmod N$. Crucial to TIDE is the term x_{-t}, where $(x_{-t})^{2^t} \equiv x_0 \bmod N$, which is derived using an extension of Eulers Criterion and the Chinese Remainder Theorem.

Now, any party wishing to solve the puzzle sequentially calculates the term $x_{-1} := x' \equiv \sqrt{x_0}$ by repeated squaring. The term x' has the property $\mathcal{J}_N(x') = +1$. This is crucial because x is chosen such that $\mathcal{J}_N(x) = -1$. Therefore, the solving party obtains the term $x^2 \equiv x'^2 \equiv x_0 \bmod N$, where $x \not\equiv x' \bmod N$. Thus, the party obtains all four square roots of x_0. Therefore, one can recover the non-trivial factors of N in polynomial time using the result from Rabin [145].
The simplicity of RSA-OEAP encryption and decryption makes TIDE a conceptually simple approach to sealed-bid auctions, whilst the underlying number theoretic techniques allow the functionality to be very efficient and practical.

6.1.3 Related Work

Alternatives to Vickrey auctions The most common style of auction is the sealed first-bid auction where the highest bidder wins and pays the amount they bid for the goods. From a cryptographic perspective this is straightforward to implement which allows additional properties to be integrated into the scheme. For example, research has been done into sealed first-bid auction schemes where the bids of losers remain hidden. This is done by requiring bidders to run a protocol computing the highest bid and ensuring that only the highest bid is opened [152, 43].

In the paper 'Secure Multiparty Computation Goes Live' [47] techniques from multi-party computation were used to implement a double auction. In a double auction sellers indicate how much of an item they are willing to sell at for certain price points. The buyers in a double auction indicate how much of the same item they are willing to buy at each price point. Using this information the *market clearing price*, which is the price per unit of an item. Calculating the market clearing price allows transactions to be made at this price point.

Both of these examples rely on a very different framework to that of TIDE. They require multiple parties being online at the same time carrying out a protocol. As such, whilst linked by the application of auctions this work is tangential to ours. More closely related to our work is the area of delay-based cryptography which we now review.

Encrypting a message to the future Time-lock puzzles, discussed in Section 4.2, were introduced to encrypt a message to the future. The method they used to build the delay is the RSW time-lock assumption, noted in Definition 1. Recently, there have been some alternative approaches to building TLPs. Rather than using repeated squaring new TLPs include using witness encryption and bitcoin [111], randomised encodings [42], and random isogeny walks over elliptic curves [75].

The RSW time-lock assumption has been used as the base of various constructions of *verifiable delay functions* (VDFs) [48, 141, 166]. In a VDF a solver computes a delay similarly to a time-lock puzzle, but rather than decrypting a message at the end, the solver instead proves that they have spent the prescribed amount of time on the computation. This proof of elapsed time has primarily found use in randomness beacons which are used in blockchain design [63].

In 2019 De Feo et al. introduced a VDF based upon isogeny walks [75], which in 2021 they extended to a *delay encryption* (DE) scheme. DE is similar to a time-lock puzzle, but rather than proving that time has elapsed, instead a *session key* is derived. This can be seen as similar to the decryption key described in the technical overview of TIDE, in Section 6.1.2. Indeed DE as a primitive is very similar to the notion of timed-release encryption where we align our TIDE construction. The key difference between the two primitives is that DE uses notions from identity-based encryption and thus avoids using a trapdoor in the setup phase.

Recall in Section 6.1.1 that timed-release encryption (TRE) is another delay-based primitive, whose traditional definition combines public-key encryption with a time-server [57, 121]. Messages can be encrypted to a public key and decryption requires a trapdoor which is kept confidential by a time-server until an appointed time. In a recent paper by Chvojka et al. [62] TRE was defined generically with a view to improving versatility and functionality. With TIDE we build a timed-release encryption scheme following the definitions of Chvojka et al. However, our scheme does not require a time-server.

6.1.4 Contributions

In our work we design a novel and theoretically efficient variant of a time-lock puzzle by utilising RSA-OEAP encryption and decryption to obtain a simple and efficient construction. We provide a security and efficiency analysis of our construction, proving that TIDE is cryptographically secure under the TRE notions [62]. We analyse the theoretical and practical efficiency of our scheme, proving that it has concrete theoretical advantages over the alternative proposals for Vickrey auctions and demonstrate that it is significantly more practical than current candidates. We present evidence of the practicality of our scheme by providing an implementation study using Raspberry Pi devices and a desktop PC, showing that TIDE can be run efficiently on consumer grade hardware. In particular, we show that when using a 2048-bit modulus, TIDE takes approximately one second to setup on a desktop PC and 30 seconds on a Raspberry Pi.

6.2 Preliminaries: Assumptions and Number Theory

In this section we review the time-lock assumption and number theory required to construct TIDE. For well known theorems we refer to the relevant sources for proofs and we prove the other theorems in Section 6.4.

The RSW time-lock assumption [151] is core to a number of notable constructions using a cryptographic delay in the latest literature [120, 79, 48, 141, 166, 74].

The definition of the RSW time-lock assumption and the related Square and Multiply Algorithm can be found in Section 4.3.3 under Definition 1 and Algorithm 1 respectively.

Next, we recall from the Fermat-Euler Theorem in Section 4.3.3, Theorem 2, and Corollary 1, that if the group order $\phi(N)$ is known, then the RSW time-lock assumption is no longer relevant. That is, by using the Fermat-Euler Theorem, the input parameter b in Algorithm 1 can be reduced by the group order.

Next we note that the modulus N used in our TIDE construction will be a Blum integer [44]. Recall from Section 4.3.3 that a Blum integer $N = pq$, is the product of two Gaussian primes. A Gaussian prime has the property $p \equiv 3 \bmod 4$.

Next, recall from Section 4.3.4, Definition 3, that a quadratic residue are numbers r that satisfy the congruence in equation 4.2.

The Jacobi symbol, denoted $\mathcal{J}_N(r)$, is a function which defines the quadratic character of r in Equation 4.2. The Jacobi Symbol can be calculated in polynomial time using Euler's Criterion, noted in Section 4.3.4, under Theorem 3.

When the modulus is a prime number if the Jacobi symbol evaluates to $+1$ then r is always a quadratic residue and if the Jacobi symbol evaluates to -1 then r is always a quadratic non-residue.

The Jacobi symbol is more complex when the modulus is a composite number $N = pq$. Corollary 2 in Section 4.3.4, states that Euler's Criterion can be used to calculate the Jacobi Symbol of the number r in Equation 4.2 for a composite modulus N if the factorisation of N is known.

In Section 4.3.4 it was shown that determining the quadratic character of r when N is composite. The quadratic character of r can be determined by using

Algorithm 3 shows how to determine the quadratic character of r for composite N using Theorem 3 and Corollary 2. When N is composite the quadratic character of r can take three formats. If the Jacobi symbol evaluates to -1 then r is always a quadratic non-residue, denoted $\mathcal{QNR}_N^{-1}$. However, if the Jacobi symbol evaluates to $+1$ then r can either be a quadratic residue, denoted $\mathcal{QR}_N$ or a quadratic non-residue denoted $\mathcal{QNR}_N^{+1}$.

Also important for our TIDE construction is the distribution of quadratic residues in the ring of integers modulo N. In Section 4.3.4 the distribution of quadratic residues is discussed in Theorem 4.

To summarise, in $\mathbb{Z}_N^*$, $\frac{1}{4}$ of the elements are quadratic residues, $\frac{1}{4}$ of the elements are quadratic residues with Jacobi symbol $+1$, and $\frac{1}{2}$ of the elements are quadratic residues with Jacobi symbol -1.

Next, we discuss how to calculate preceding terms of the seed term $x_0 \in \mathcal{QR}_N$ in an RSW time-lock sequence. By way of comparison, if we wish to calculate the subsequent term of x_0 in an RSW sequence we evaluate $x_1 \equiv x_0^{2^1} \bmod N$ by inputting $(x_0, 2^1, N)$ into Algorithm 1.

In contrast if we wish to calculate the preceding term of x_0, that is the term x_{-1}, in polynomial time we must know the factorisation of N. If the factorisation of N is known Theorem 3 can be used in conjunction with the Chinese Remainder Theorem (CRT) to calculate the term x_{-1} in polynomial time. The discussion regarding moving in the *backward* direction of an RSW time-lock sequence can be found in Section 4.3.5 and the

CRT can be found in Theorem 7.

The next important property for our TIDE construction is that if $r \in \mathcal{QR}_N$ then the CRT implies that there are four distinct solutions to Equation 4.2. The details of $r \in \mathcal{QR}_N$ having four unique solutions is discussed extensively in Section 4.4.1.

Of further interest to our TIDE construction, is that if N is a Blum integer, then the four square roots of each $r \in \mathcal{QR}_N$ has specific properties. That is, two of the square roots of r are quadratic non-residues with Jacobi symbol -1, one square root is a quadratic non-residue with Jacobi symbol $+1$, and one square root is a quadratic residue.

Theorem 18. Let N be a Blum integer. Then for all $r \in \mathcal{QR}_N$, if $x^2 \equiv x'^2 \equiv r \bmod N$, where $x \neq \pm x'$, then without loss of generality $\mathcal{J}_N(\pm x) = -1$, and $\mathcal{J}_N(\pm x') = +1$. That is $\pm x \in \mathcal{QNR}_N^{-1}$, $x' \in \mathcal{QR}_N$ and $-x' \in \mathcal{QNR}_N^{+1}$. We refer to $x' \in \mathcal{QR}_N$ as the principal square root of $r \bmod N$.

Proof. If N is a Blum integer, then $N \equiv 1 \bmod 4$. By Theorem 9 every $x_0 \in \mathcal{QR}_N$ has four distinct square roots $\pm x$ and $\pm x'$. As $N \equiv 1 \bmod 4$, by the Law of Quadratic Reciprocity, in Theorem 5, $\mathcal{J}_N(x) = \mathcal{J}_N(-x)$ and $\mathcal{J}_N(x') = \mathcal{J}_N(-x')$. It must be the case that $x^2 \equiv x'^2 \bmod N$, which implies $(x-x')(x+x') \equiv 0 \bmod N$, which implies $(x-x') \mid N$ and $(x+x') \mid N$. That is, without loss of generality $(x-x') = k \cdot p$ and $(x+x') = \ell \cdot q$, where $k, \ell \in \mathbb{N}$. Therefore, $\mathcal{J}_p(x) = \mathcal{J}_p(x')$ and $\mathcal{J}_q(x) = \mathcal{J}_q(-x')$. As $p \equiv 3 \bmod 4$, the law of quadratic reciprocity tells us $\mathcal{J}_p(-1) = -1$, we have $\mathcal{J}_q(x) \cdot \mathcal{J}_p(-1) = \mathcal{J}_q(-x') \cdot \mathcal{J}_p(-1)$. This implies that $\mathcal{J}_N(-x) = \mathcal{J}_N(x')$ or written another way $\mathcal{J}_N(x) \neq \mathcal{J}_N(x')$.

Without loss of generality, eliminate the two roots with $\mathcal{J}_N$ equal to -1, say $\mathcal{J}_N(x) = \mathcal{J}_N(-x) = -1$. This leaves $\mathcal{J}_N(x') = \mathcal{J}_N(-x') = +1$. It is the case that only one of $-x'$ or x' has $\mathcal{J}_p = \mathcal{J}_q = 1$ as $p \equiv 3 \bmod 4$. Therefore, without loss of generality, it is only x' that has the property $\mathcal{J}_N(x') = +1$ and $x' \in \mathcal{QR}_N$ [44]. □

Finally, we discuss a method to factor a Blum integer N in polynomial time if specific information is provided.

Fermat's factorisation method is a technique to factor an odd composite number $N = pq$ in exponential time [68]. The method requires finding x and x' such that $x^2 - x'^2 = N$ is satisfied. Then the left-hand side can be expressed as a difference of squares $(x - x')(x + x') = N$.

Fermat's method can be extended to finding x and x' to satisfy the following weaker congruence of squares condition $x^2 \equiv x'^2 \bmod N$, where $x \not\equiv \pm x'$. This congruence can be expressed as $(x - x')(x + x') \equiv 0 \bmod N$. Finding a congruence of squares forms the basis for several sub-exponential sieving-based factorisation algorithms [68]. However, if x and x' in a congruence of squares are known, then factoring N can be done in polynomial time.

Theorem 19. Let N be a Blum integer. If x and x' are known such that $x^2 \equiv x'^2 \bmod N$, where $x \not\equiv \pm x' \bmod N$, then the non-trivial factors of N can be recovered in polynomial time.

The definition of Fermat's factoring method and the weaker Congruence of Squares condition can be found in Section 4.4.2, in Definition 6 and Definition 7 respectively.

Proof. Proof for Theorem 19 can be found in Section 6.4. □

6.3 Our Construction

In this section we give the concrete details of our construction TIDE. Formally, TIDE is a TRE scheme and we provide a formal exposition of its security properties in Section 6.4. In our TRE scheme $\mathcal{C}$ is the Challenger, $\mathcal{S}$ is the Solver, and $\mathcal{A}$ is the Adversary. In the context of Vickery auctions, $\mathcal{C}$ can be thought of as the auctioneer and $\mathcal{S}$ can be thought of as a bidder. As is customary, multiple bidders are participate in an auction.

A TRE scheme consists of four algorithms: Gen, Solve, $\mathsf{Encrypt}$, $\mathsf{Decrypt}$. Gen and Solve provide the time-lock element of the scheme: Gen generates a secret key, public key and a puzzle, Solve takes the puzzle and recovers the corresponding secret key. $\mathsf{Encrypt}$ can be ran by multiple parties (bidders) simultaneously using the public key. Solve can be run by any party i.e., any bidder or third party can run this algorithm. Once Solve has terminated, the Solver can then use the secret key to decrypt all of the bids encrypted with $\mathsf{Encrypt}$ by using the $\mathsf{Decrypt}$ algorithm. We now outline the details of the four TIDE algorithms.

- $(\mathsf{sk}, \mathsf{pk}, P, t) \leftarrow \mathsf{Gen}(1^\kappa, t)$ takes as input a security parameter 1^κ and time parameter t and outputs a secret key sk, public key pk, puzzle P, and time parameter. The secret key consists of the factors of $\mathsf{sk} := (p, q)$ and the public key consists of an RSA modulus N and fixed encryption exponent $e := 2^{16} + 1 = 65537$. The puzzle is set to $P := (x, x_0, x_{-t})$, where $x^2 \equiv x_0 \bmod N$, $\mathcal{J}_N(x) = -1$, and where $(x_{-t})^{2^t} \equiv x_0 \bmod N$.

- $\mathsf{sk} \leftarrow \mathsf{Solve}(\mathsf{pk}, P, t)$ takes as input the public key pk, puzzle P, and time parameter t and outputs the secret key $\mathsf{sk} := (p, q)$, where $N = pq$.
- $c \leftarrow \mathsf{Encrypt}(\mathsf{pk}, m)$ takes as input a public key $\mathsf{pk} := (N, e)$ and a message m and outputs a ciphertext c.
- $\{m, \perp\} \leftarrow \mathsf{Decrypt}(\mathsf{sk}, c)$ takes as input the secret key $\mathsf{sk} := (p, q)$ and a ciphertext c as input and outputs a message m or error $\perp$.

Algorithm 15: Gen run on security parameter 1^κ and time parameter t to create the secret key sk, public key pk and puzzle P.

```
   input  : $1^\kappa, t$
1  $p, q := 1$
2  while $p = q$ do
3      $p := \mathsf{prime}(\frac{\kappa}{2})$
4      $q := \mathsf{prime}(\frac{\kappa}{2})$
5  end
6  $N := pq$
7  $\mathcal{J}_p(x), \mathcal{J}_q(x) := 1$
8  while $\neg(\mathcal{J}_p(x) = 1 \wedge \mathcal{J}_q(x) \neq 1) \wedge \neg(\mathcal{J}_p(x) \neq 1 \wedge \mathcal{J}_q(x) = 1)$ do
9      $x := \mathcal{U}(2, N)$
10     $\mathcal{J}_p(x) := x^{\frac{p-1}{2}} \bmod p$
11     $\mathcal{J}_q(x) := x^{\frac{q-1}{2}} \bmod q$
12 end
13 $x_0 := x^2 \bmod N$
14 $\alpha_t := x_0^{\frac{p+1}{4}^t \bmod p-1} \bmod p$
15 $\beta_t := x_0^{\frac{q+1}{4}^t \bmod q-1} \bmod q$
16 $x_{-t} := \alpha_t q(q^{-1} \bmod p) + \beta_t p(p^{-1} \bmod q) \bmod N$
17 $P := (x, x_0, x_{-t})$
   output: $(\mathsf{sk}, \mathsf{pk}, P, t)$
```

1) $\mathcal{C}$ runs $(\mathsf{sk}, \mathsf{pk}, P, t) \leftarrow_{\mathrm{R}} \mathsf{Gen}(1^\kappa, t)$ to generate the secret key, public key, and puzzle as seen on Algorithm 15 Gen. The function $\mathsf{prime}(j)$ on lines 3 and 4 is the Miller-Rabin Monte Carlo algorithm [126] which generates j bit Gaussian primes. That is, $p \leftarrow_{\mathrm{R}} \mathsf{prime}(j)$. This guarantees that N, which is calculated on line 6, is a Blum integer. Gen then enters a while loop. The purpose of the while loop is to find an x such that $x \in \mathcal{QNR}_N^{-1}$. The logic statement on line 8 condenses the conditional statements in lines 3, 5 and 7 of Algorithm 3 using De Morgan's laws [88]. Once a suitable x is found

x_0 is set to $x^2 \bmod N$. Once x is sampled and x_0 is computed the term x_{-t} is calculated, where $(x_{-t})^{2^t} \equiv x_0 \bmod N$. To calculate x_{-t} in polynomial time, Euler's Criterion, the Fermat-Euler Theorem, and the Chinese Remainder Theorem (CRT) must be applied.

Next, α_t is calculated, where α_t is the t^{th} square root of $x_0 \bmod p$. To complete the calculation of the term x_{-t}, the CRT is used on line 16, where the terms $(q^{-1} \bmod p)$ and $(p^{-1} \bmod q)$ are calculated using Euclid's Extended Algorithm (EEA). Theorem 8 tells us that $\alpha \equiv \sqrt{x_0} \equiv x_0^{\omega} \bmod p$, where $\omega = \frac{p+1}{4}$. Let α_t be the t^{th} square root of $x_0 \bmod p$. For example, if $t = 2$, then $\alpha_2 \equiv \sqrt{\sqrt{x_0}} \equiv (x_0^{\omega})^{\omega} \equiv x_0^{\omega^2}$. Therefore, $\alpha_t \equiv x_0^{\omega^t} \bmod p$. Note that the exponent ω^t, for large t will make calculating $x_0^{\omega^t} \bmod p$ computationally infeasible. Therefore, the Fermat-Euler Theorem is used so the exponent ω^t can be reduced $\bmod (p-1)$. Next, β_t is calculated, where β_t is the t^{th} square root of $x_0 \bmod q$. β_t is calculated in a similar fashion as α_t, except ω is set to $\frac{q+1}{4}$ and ω^t is reduced $\bmod (q-1)$.

The puzzle P is set to the tuple (x, x_0, x_{-t}) and then $\mathcal{C}$ securely stores sk and passes (pk, P, t) to $\mathcal{S}$ who must solve:

$$\text{Given } (\mathsf{pk} \coloneqq (N, e), P \coloneqq (x, x_0, x_{-t}), t), \text{ find the factors of } N.$$

2) $\mathcal{S}$ (or any party) runs $\mathsf{sk} \leftarrow \mathsf{Solve}(\mathsf{pk}, P, t)$ to solve the challenge, as seen on Algorithm 16 Solve. First Solve calculates the term x' in $t-1$ sequential steps by evaluating $(x_{-t})^{2^{t-1}} \bmod N$. This is where the sequential calculation takes place using Algorithm 1 with inputs $(x_{-t}, 2^{t-1}, N)$. The term x' is guaranteed to be in $\mathcal{QR}_N$ by Definition 3. $\mathcal{S}$ now has $x \in \mathcal{QNR}_N^{-1}$ and $x' \in \mathcal{QR}_N$. Therefore, x must be distinct from x', and we have $x^2 \equiv x'^2 \equiv x_0 \bmod N$. Finally, using the result from Theorem 19, Solve calculates $\mathsf{gcd}(x - x', N)$ to recover one factor p' of N using Euclid's Extended Algorithm. Next, $\frac{N}{\mathsf{gcd}(x-x', N)}$ is calculated to recover the other factor q'.

Algorithm 16: Solve runs on the public key, puzzle, and time parameter pk, P, t to recover the secret key sk.

```
  input : $\mathsf{pk} \coloneqq (N, e), P = (x, x_0, x_{-t}), t$
1 $x' \coloneqq (x_{-t})^{2^{t-1}} \bmod N$
2 $p' \coloneqq \mathsf{gcd}(x - x', N)$
3 $q' \coloneqq \frac{N}{p'}$
4 $\mathsf{sk} \coloneqq (p', q')$
  output: $\mathsf{sk}$
```

3) $\mathcal{S}$ runs $c \leftarrow \mathsf{Encrypt}(\mathsf{pk}, m)$ as seen in Algorithm 17 Encrypt. Encrypt inputs the public key $\mathsf{pk} \coloneqq (N, e)$ and encrypts a message m using RSA-OEAP encryption and

Algorithm 17: Encrypt runs on a message public key pk and message m, to produce ciphertext c.

```
input : $\mathsf{pk} := (N, e), m$
1 $k_0, k_1, G, H \leftarrow \mathsf{params}(1^\kappa)$                  // OAEP parameters
2 $m' := m \mid\mid 0^{k_1}$                                  // Zero pad to $n - k_0$ bits
3 $r := \text{rand}(k_0)$                        // Generate a random $k_0$ bit number
4 $X := m' \oplus G_{n-k_0}(r)$                          // Hash $r$ to length $n - k_0$
5 $Y := r \oplus H_{k_0}(X)$                              // Hash $X$ to length $k_0$
6 $m'' := X \mid\mid Y$                                  // Create message object
7 $c := m''^e \bmod N$                                          // RSA encrypt
output: c
```

outputs the ciphertext c. First Encrypt outputs the RSA-OAEP parameters k_0, k_1, G, H, where k_0 and k_1 are constants used for padding and G and H are hashing algorithms modelled as random oracles. Using RSA-OAEP, parties can encrypt messages to this modulus and encryption exponent. This means that messages can only be decrypted using the Decrypt algorithm only after Solve has recovered the secret key sk.

Note that the Solve and Encrypt algorithms are not sequential. The Encrypt algorithm can be run by any Solver (Bidder) using pk prior to the Solve algorithm recovering the sk.

4) $\mathcal{S}$ runs $\{m, \perp\} \leftarrow \mathsf{Decrypt}(\mathsf{sk}, c)$ as seen in Algorithm 18 Decrypt. Decrypt inputs the secret key $\mathsf{sk} := (p, q)$ and decrypts ciphertext c using RSA-OEAP encryption and recovers the message m or outputs an error $\perp$. Decrypt also outputs the same RSA-OAEP parameters k_0, k_1, G, H as Encrypt. Next, Decrypt recovers the decryption exponent d on lines $2, 3, 4$, where Euclids Extended Algorithm is used. Finally, the RSA-OEAP decrypt algorithm removes the padding and randomness added during the encryption to recover the message m.

In addition, we introduce an alternative method to sample the seed for the challenge through the optional Algorithm 19 named Samplex. This algorithm is designed to efficiently sample the parameter x to be used to generate the puzzle P. The selection of the parameter x itself does not require the knowledge of the factors p and q of the modulus N. However, knowledge of these factors will be required to generate the puzzle.

The Samplex algorithm selects x from $\mathbb{Z}_N^*$ in a uniform manner. Furthermore, it employs the Jacobi algorithm, with details noted in Section 4.3.4, Algorithm 4.

By leveraging the approach of using the Samplex algorithm, the task of seeding the parameter x within the puzzle becomes accessible to *any* party. The Samplex algo-

Algorithm 18: Decrypt runs on secret key sk and ciphertext c, to produce message m.

```
   input  : sk := $(p', q'), c$
1  $k_0, k_1, G, H \leftarrow \mathsf{params}(1^\kappa)$          // OAEP parameters
2  $N := p'q'$
3  $\phi(N) := (p'-1)(q'-1)$
4  $d := e^{-1} \bmod \phi(N)$          // recover $d$ using EEA
5  $m'' := c^d \bmod N$
6  $X := \lfloor c'' \cdot 2^{-k_0} \rfloor$          // Extract $X$
7  $Y := m'' \bmod 2^{k_0}$          // Extract $Y$
8  $r := Y \oplus H_{k_0}(X)$          // Recover $r$
9  $m' := X \oplus G_{n-k_0}(r)$          // Recover padded message
10 $m := m' \cdot 2^{-k_1}$          // Remove padding
   output: m
```

rithm relies on the context provided in Chapter 4, Definition 4 regarding the Quadratic Residuosity Problem.

The incorporation of Algorithm 19 offers a viable alternative to the while loop in lines 7–12 of Algorithm 15 Gen. The proof of correctness of Algorithm 19 can be found in Bach et al. [29]. Therefore, the proof of Corollary 4 is also relevant in the case of Algorithm 19. The original Python code for the function Jacobi(x, pk) can be found in [65].

Algorithm 19: Samplex is run on public key pk to efficiently sample parameter x without knowledge of p and q.

```
   input  : pk
1  while $j \neq -1$ do
2      $x := \mathcal{U}(2, pp)$
3      $j :=$ Jacobi$(x, \mathsf{pk})$
4  end
   output: $x$
```

6.4 Security

We provide a security analysis of our construction. We first recall the formal definition of timed-release encryption (TRE), following Chvojka et al. [62][1], as well as the definitions

[1]In [62] they offer a generalised version of this definition, to incorporate what they define *sequential timed-release encryption*. This is beyond the scope of this work, and we instead specify the "non-

of correctness and security for a TRE scheme. The symbol $\leftarrow$ indicates deterministic evaluation and the symbol $\xleftarrow{R}$ indicates probabilistic evaluation.

Definition 10. A timed-release encryption scheme with message space $\mathcal{M}$ is a tuple of algorithms TRE = (Gen, Solve, Encrypt, Decrypt) defined as follows.

- $(\mathsf{pk},\mathsf{sk},P,t) \xleftarrow{R} \mathsf{Gen}\left(1^\kappa, t\right)$ is a probabilistic algorithm which takes as input a security parameter 1^κ and a time hardness parameter t, and outputs a public encryption parameter pk, a secret key sk, and a puzzle P. We require that Gen runs in time $\text{poly}\left((\log t), \kappa\right)$.
- $\mathsf{sk} \leftarrow \mathsf{Solve}(\mathsf{pk}, P, t)$ is a deterministic algorithm which takes as input a public key pk, a puzzle P, and a time parameter t, and outputs a secret key sk. We require that Solve runs in time at most $t \cdot \mathsf{poly}(\kappa)$.
- $c \xleftarrow{R} \mathsf{Encrypt}\left(\mathsf{pk}, m\right)$ is a probabilistic algorithm that takes as input public encryption parameter pk and message $m \in \mathcal{M}$, and outputs a ciphertext c.
- $m/\perp \leftarrow \mathsf{Decrypt}\left(\mathsf{sk}, c\right)$ is a deterministic algorithm which takes as input a secret key sk and a ciphertext c, and outputs $m \in \mathcal{M}$ or $\perp$.

Definition 11 (Correctness)**.** A TRE scheme is *correct* if for all $\kappa \in \mathbb{N}$ and hardness parameter t, it holds that

$$\Pr\left[m = m' : \begin{array}{r} (\mathsf{pk},\mathsf{sk},P,t) \xleftarrow{R} \mathsf{Gen}\left(1^\kappa, t\right), \mathsf{sk} \leftarrow \mathsf{Solve}\left(\mathsf{pk}, P, t\right) \\ m' \leftarrow \mathsf{Decrypt}\left(\mathsf{sk}, \mathsf{Encrypt}\left(\mathsf{pk}, m\right)\right) \end{array} \right] = 1$$

Definition 12 (Security)**.** A timed-release encryption scheme is secure with gap $0 < \epsilon < 1$ if for all polynomials n in κ there exists a polynomial $\tilde{t}(\cdot)$ such that for all polynomials t fulfilling that $t(\cdot) \geq \tilde{t}(\cdot)$, and every polynomial-size adversary $\mathcal{A} = \{(\mathcal{A}_{1,\kappa}, \mathcal{A}_{2,\kappa})\}_{\kappa\in\mathbb{N}}$ there exists a negligible function $\mathsf{negl}(\cdot)$ such that for all $\kappa \in \mathbb{N}$ it holds

$$\mathbf{Adv}^{\mathrm{TRE}}_{\mathcal{A}} = \left| \Pr\left[\begin{array}{r} \mathsf{pk}, P \xleftarrow{R} \mathsf{Gen}\left(1^\kappa, t\right) \\ (m_0, m_1, \mathrm{st}) \leftarrow \mathcal{A}_{1,\kappa}(\mathsf{pk}, P) \\ b = b' : b \xleftarrow{s} \{0,1\}; c \xleftarrow{R} \mathsf{Encrypt}\left(\mathsf{pk}, m_b\right) \\ b' \leftarrow \mathcal{A}_{2,\kappa}(c, \mathrm{st}) \end{array} \right] - \frac{1}{2} \right| \leq \mathsf{negl}(\kappa)$$

It is required that $|m_0| = |m_1|$ and that the adversary $\mathcal{A}_\kappa = (\mathcal{A}_{1,\kappa}, \mathcal{A}_{2,\kappa})$ consists of two circuits with total depth at most $t^\epsilon(\kappa)$ (i.e., the total depth is the sum of the depth of $\mathcal{A}_{1,\kappa}$ and $\mathcal{A}_{2,\kappa}$). Furthermore, both $\mathcal{A}_{1,\kappa}$ and $\mathcal{A}_{2,\kappa}$ have knowledge of m_0 and m_1.

sequential" case.

In what follows, we will refer to algorithms 'taking t time to compute', and 'bounding computation time by t'. In both cases, we are referring to evaluating a polynomial sized arithmetic circuit of depth at most t.

In order to prove the security of TIDE, we must first define a new hardness assumption. Informally, this states that the terms x, x_0 and x_{-t} provide a negligible advantage to factoring a Blum integer N, or distinguishing between ciphertexts encrypted to a Blum integer N, when the computational time is bounded by t.

Definition 13 (BBS Shortcut Assumption)**.** Let the RSA Assumption be that for any $N \leftarrow_{\mathrm{R}} \mathsf{Gen}\left(1^{\kappa}\right)$ and $e = 65537$, it is hard for any probabilistic polynomial-time algorithm to find the e-th root modulo N of a random $y \leftarrow_{\mathrm{R}} \mathbb{Z}_N^*$ [150].

The BBS Shortcut Assumption states that given (N', e) and terms (x, x_0, x_{-t}), where $N' \leftarrow_{\mathrm{R}} \mathsf{Gen}\left(1^{\kappa}\right)$ is a randomly sampled Blum integer, $e = 65537$, x is a randomly sampled integer such that $x \in \mathcal{QNR}_N^{-1}$, $x_0 := x^2 \bmod N$, and x_{-t} is the term $t+1$ steps before x_0 in a BBS_CSPRNG sequence, it is no easier to find the e-th root of a random $y' \leftarrow_{\mathrm{R}} \mathbb{Z}_{N'}^*$ than to find the e-th root modulo N of a random $y \leftarrow_{\mathrm{R}} \mathbb{Z}_N^*$ in a standard RSA instance, without first factoring N'.

We now analyse this security assumption, in order to relate it to the RSA assumption that RSA with OAEP relies on [81].

Recall $P = (x, x_0, x_{-t})$ consists of a randomly sampled integer x, and two terms x_0, x_{-t} which by construction are part of the BBS-CSPRNG sequence, and hence are pseudorandom. As we will see in Lemma 2, the relation between these integers exactly relates to the evaluation of the BBS-CSPRNG sequence, which allows N' to be factored, and cannot be evaluated in time less than t, for some $t \in \mathbb{N}$. The crux of the assumption is that x_{-t} is only related to the terms x and x_0 by the repeated squaring property, which allows the Blum integer N' to be factored. By the RSW time-lock assumption, we know that this will take t time to evaluate, and hence we assume that $P = (x, x_0, x_{-t})$ are only useful when factoring N'

We now prove TIDE is a timed-release encryption scheme satisfying correctness and security.

Theorem 20. TIDE is correct.

Proof. First, consider the following statement:

For any message $m \in \{0,1\}^*$, $\mathsf{Decrypt}\Big(\mathsf{Encrypt}\big(N, m\big), (p, q)\Big)$ outputs m, where $\mathsf{Encrypt}$ and $\mathsf{Decrypt}$ are described in Algorithms 17 Encrypt and Algorithm 18 Decrypt respectively. This corresponds to the statement that the RSA cryptosystem with OAEP is correct, which is known to be true [81].

Now suppose Algorithm 15 Gen has been run, such that the following parameters have been generated: a public key N, puzzle $P = (x, x_0, x_{-t})$ and time parameter t, and a secret key $\mathsf{sk} = (p, q)$. What remains is to prove that Solve outputs $\mathsf{sk} = (p, q)$. This proof will require a sequence of arguments based on the Theorems outlined in Section 6.2.

First we must prove that Algorithm 15 Gen correctly selects the term x such that $x \in \mathcal{QNR}_N^{-1}$. □

Corollary 4. (Of Theorem 4). The while loop on lines 8-12 of Algorithm 15 Gen selects $x \in \mathcal{QNR}_N^{-1}$ with overwhelming probability.

Proof. The while loop on lines 8-12 of Algorithm 15 Gen selects a quadratic non-residue with Jacobi Symbol equal to -1 by running a series of Bernoulli trials with probability $\mathrm{P}\left(x = \mathcal{QNR}_N^{-1}\right) = \frac{1}{2}$. This forms a geometric distribution $G \sim \mathrm{Geo}(\frac{1}{2})$. Therefore, we can expect to find $x \in \mathcal{QNR}_N^{-1}$ in $\mathbb{E}\{G\} = 2$ trials. □

Second we prove that Algorithm 15 Gen correctly calculates the term x_{-t}, which is the t^{th} principal square root of x_0. Proof that Algorithm 15 Gen correctly calculates the term x_{-t} first relies on the proof of Theorem 8, which can be found in Section 4.4.1. Theorem 8 tells us that we can take square roots of any quadratic residue of a Gaussian prime by calculating $\alpha \equiv r^{\frac{p+1}{4}} \bmod p$.

Next, we prove that Gen correctly calculates the t^{th} principal square root of x_0.

Theorem 21. The Algorithm 15 Gen correctly calculates the t^{th} principal square root x_{-t} of the seed x_0.

Proof. Let $\omega = \frac{p+1}{4}$. If Algorithm 15 Gen provides the seed term $x_0 \in \mathcal{QR}_N$, then, by Theorem 8, the t^{th} principal square root of $x_0 \bmod p$ is $\alpha_t := x_0^{\omega^t} \bmod p$ and the t^{th} principal square root of $x_0 \bmod q$ is $\beta_t := x_0^{\omega^t} \bmod q$. Then, the Chinese Remainder Theorem 7 is used to calculate:
$x_{-t} := [\alpha_t q (q^{-1} \bmod p) + \beta_t p (p^{-1} \bmod q)] \bmod N$. □

Third we must prove that the Algorithm 16 Solve correctly calculates the term $x' \in \mathcal{QR}_N$ using Algorithm 1. This follows from Theorem 1, which demonstrates that Algorithm 1 correctly calculates the term x_i in a BBS-CSPRNG sequence.

Finally, Theorem 19 is proven to show that Algorithm 16 Solve calculates $\mathsf{gcd}(x' - x, N)$ to recover a non-trivial factor of N [145].

Proof. (Theorem 19.) As x and x' are distinct we have $x^2 \equiv x'^2 \bmod N$. This implies that $pq \mid x^2 - x'^2$. As p and q are both prime this indicates that $p \mid (x - x')(x + x')$ and $q \mid (x - x')(x + x')$. Also, because p is prime it must be the case that $p \mid (x - x')$ or $p \mid (x + x')$. Similarly, it must be the case that $q \mid (x - x')$ or $q \mid (x + x')$. Without loss of generality, assume that $p \mid (x - x')$ is true and that $q \mid (x - x')$ is true. This implies that $pq \mid (x - x')$, which indicates that $x \equiv x' \bmod N$. This is a contradiction because x and x' are distinct. Then it must be the case that $p \mid (x - x')$ and $q \nmid (x - x')$. Therefore, one of the factors of N can be recovered by calculating $p' := \mathsf{gcd}(x - x', N)$ using Euclid's Extended Algorithm, and the other factor of N can be recovered by calculating $q' := \frac{N}{\mathsf{gcd}(x - x', N)} = \frac{N}{p'}$. □

We now prove that Solve outputs $\mathsf{sk} = (p, q)$, and hence Theorem 20: the correctness of TIDE.

Proof. (Theorem 20) For any pk, sk, and puzzle generated by Gen, we show that sk can be recovered by Solve. More precisely, let $N = pq, P := (x, x_0, x_{-t}), t$ be output by Gen, before being input into Algorithm 16 Solve. Algorithm 16 Solve will calculate the term x' by entering the following parameters $(x_{-t}, 2^{t-1}, N)$ into Algorithm 1, which will output $x' := (x_{-t})^{2^{t-1}} \bmod N$. The term x' is guaranteed to be correct by Theorem 1 and is guaranteed to be in $\mathcal{QR}_N$ by Definition 3, and hence we have that $x \in \mathcal{QNR}_N^{-1}$ and $x' \in \mathcal{QR}_N$. This guarantees that x must be distinct from x'. Therefore, by Theorem 19, calculating $p' = \mathsf{gcd}(x - x', N)$ will recover one factor of N using Euclid's Extended Algorithm, and the other factor can be recovered by calculating $q' = \frac{N}{\mathsf{gcd}(x - x', N)}$.

□

Theorem 22. TIDE is a secure TRE scheme under the RSW time-lock assumption and the RSA and BBS-shortcut assumptions.

To prove TIDE secure, we show that two messages encrypted using public key (N, e) are indistinguishable under a chosen plaintext attack, where the adversary is bounded by t computation time. We first note that the underlying encryption scheme is RSA with OAEP padding, which is IND-CPA secure [81]. In our proof we provide a reduction from the TRE security of TIDE to IND-CPA security of RSA with OAEP. Explicitly, this requires proof that giving an adversary the additional parameters of P and t, and bounding their computation time by t offers a negligible advantage over the standard RSA-OAEP game.

We first prove the following statement.

Lemma 2. Given any (N, P, t) output by Algorithm 15 Gen, the RSA modulus N cannot be factored in time less than t, with more than negligible probability.

Proof. Let N be a random Blum integer and P be a puzzle output by Algorithm 15 Gen. Note from Algorithm 15 that $P = (x, x_0, x_{-t})$, where $x \in \mathcal{QNR}_N^{-1}$, $x_0 \equiv x^2 \bmod N$, and x_{-t} is the t^{th} square root of x_0. To factor N in time less than t, a pair of integers (p^*, q^*) must be computed, such that $p^* \neq 1, q^* \neq 1$, and $p^* q^* = N$, in less than t sequential steps.

We split the proof into two parts: i) Attempts to compute an x', where $x' \equiv \sqrt{x_0} \bmod N$ and $x' \in \mathcal{QR}_N$, in less than t sequential steps, and ii) Attempts to recover the non-trivial factors of N using a method that does not use x'.

We start by proving part (i): that computing x' in time less than t reduces to the RSW time-lock assumption. Specifically, if Solve is honestly run, then $x' := x_0^{2^{t-1}} \bmod N$ is calculated using Algorithm 1 with the input $(x_{-t}, 2^{t-1}, N)$. By the RSW time-lock assumption calculating x' using Algorithm 1 requires $t-1$ sequential steps. Once x' is calculated, Algorithm 16 Solve recovers the factors of N by calculating $p' := \mathsf{gcd}(x - x', N)$ and $q' = \frac{N}{p'}$.

Next, suppose there exists a PPT algorithm $\mathcal{E}_{<t}$ to evaluate x' in less than $t-1$ sequential steps. Finding such an x' using $\mathcal{E}_{<t}$ reduces to the RSW time-lock assumption and we obtain a contradiction. Therefore, it is not possible to recover $p^* := \mathsf{gcd}(x - x', N)$ without sequentially evaluating x' with non-negligible probability.

Next, we prove part (ii): that factoring N faster than sequential squaring reduces to an open problem. First note that N is a Blum integer, which is an RSA modulus that is the product of Gaussian primes. Therefore, we assume N cannot be factored by any PPT algorithm with more than negligible probability.

Next, giving $\mathcal{A}$ either (N, x, x_0, t) or (N, x_{-t}, t) also reduces to a standard factoring assumption, as seen in Section 4 of Rabin [145]. What remains is to show that giving an adversary all of the puzzle P does not allow them to factorise N. To see this, note that x_0 can be trivially obtained from x, and that by construction x_{-t} and x_0 are terms in a BBS_CSPRNG sequence [44]. Knowledge of these terms does not allow factorisation of N faster than sequential squaring unless $(x_{-t})^{2^{\lambda(\lambda(N))}}$ mod N is calculated efficiently. This is an open problem given by Theorem 9 of Blum et al. [44, 80, 91].

Therefore, the only way a PPT algorithm could factorise N given (pk, P, t) with non-negligible probability is to sequentially evaluate x' and subsequently recover the factors by calculating $p' := \mathsf{gcd}(x - x', N)$ and $q' = \frac{N}{p'}$. $\square$

We now use this result to obtain a reduction from the TRE security of TIDE to the standard RSA IND-CPA security.

Proof. (Theorem 10) We start by assuming that there exists an adversary $\mathcal{A} = (\mathcal{A}_{1,\kappa}, \mathcal{A}_{2,\kappa})$ who can gain a non-negligible advantage in the $\mathbf{Adv}_{\mathcal{A}}^{\mathrm{TRE}}$ game defined in Definition 12.

We use Lemma 2 and Definition 13 to show that if the adversary wins the game by factoring N we obtain a contradiction based on the RSW assumption, and if they win the game without factoring N, we obtain a contradiction based on the RSA and BBS-shortcut assumptions.

Recall from Lemma 2 that if the adversary $\mathcal{A}$ factors a Blum integer N output by Algorithm 15 in time less than t with more than negligible probability, then the RSW time-lock assumption is broken, and hence we have a contradiction.

Now, recall that RSA with OAEP padding is IND-CPA-secure under the RSA assumption [81]. Suppose $\mathcal{A}$ gained a non-negligible advantage in the TRE security game without factoring. As the underlying encryption scheme is IND-CPA secure, to distinguish between the messages m and m' with any advantage would require decrypting one of the messages, and hence taking an e-th root modulo N.

By the BBS shortcut assumption presented in Definition 13, any adversary who gains an advantage in the TRE security game could also gain the same non-negligible advantage in the standard IND-CPA game for RSA-OAEP, and hence break the RSA assumption. This gives us another contradiction. Therefore TIDE is secure under the RSW, RSA, and BBS-shortcut assumptions.

$\square$

6.5 Practicality and Implementation

In this section we detail why TIDE is the most practical solution to the issue in sealed-bid auctions detailed in the introduction when compared to previous solutions. Recall that the purpose of using time-lock puzzles in the context of auctions is to ensure that when a party does not reveal their bid it can be recovered at some computational cost by solving the relevant puzzle. The problem with this approach is that it does not scale well as the number of bidders increases. The recent works by Malavolta et al. [120] and

Burdges et al. [53] each introduce a method to ensure that only one long computation needs to be solved, regardless of the number of bidders. TIDE shares this property, as Solve only needs to be run once per auction, and as such is more efficient than the standard TLP approach.

Malavolta et al. advocate using homomorphic time-lock puzzles, to determine homomorphically which is the winning puzzle (and the second highest bid in the case of Vickrey auctions) and then solve only the relevant puzzles. The problem with this approach is that it relies on fully homomorphic encryption, or on indistinguishability obfuscation (IO), both of which are currently lacking a practical construction.

Burdges et al. propose using long chains of isogenies to achieve delay encryption, which is a more practical approach than using homomorphic time-lock puzzles. However, their candidate construction has implementation issues in the form of a large evaluator storage requirement and a setup that grows proportionally to the time delay. Whilst promising in its approach, these challenges make this candidate problematic in practice.

TIDE overcomes these issues with a setup that takes on average 1 to 2 seconds on a consumer-grade desktop PC for a 2048-bit RSA modulus. Once the public key parameters are output the delay is completely and predictably adjustable with the time parameter t. Furthermore, the maximum storage size of any party is bounded by the size of the RSA modulus.

Finally, we support these arguments with an implementation study, including code that can be run quickly on consumer-grade hardware. In the next section we provide the timings and results of our implementation analysis.

6.5.1 Implementation

In this section we describe the implementation and performance analysis of our TIDE construction. The software implementation is written in Python 3 and the code is publicly available at `https://github.com/wsAJMYbR/tide.git`.

Our testing platform consisted of two different hardware environments: a Raspberry Pi cluster, and a desktop PC. The Pi cluster consisted of four Raspberry Pi 3 Model B computers networked together. Each Pi node utilises a quad-core 1.2 GHz CPU, with 1 GB of available memory. This enabled us to run experiments on four different modulus sizes in parallel. The use of Raspberry Pi devices provides an affordable and ubiquitously available device with a consistent configuration. This facilitates the replication of our experiments and comparison with other delay-based schemes. Furthermore, as Rasp-

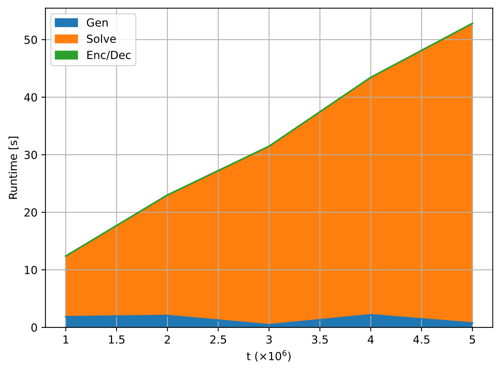

Figure 6.1: The impact of adjusting parameter t on the run time of Gen, Solve, Encrypt, and Decrypt algorithms when run on a desktop PC with a 2048 bit modulus. The primary effect is on Solve, which displays a linear increase.

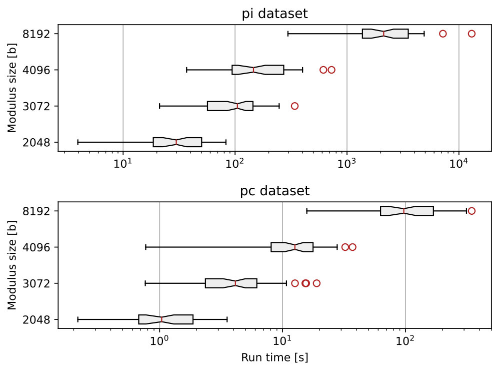

Figure 6.2: The spread of setup time across modulus sizes and machines. Setup time increases in response to an increase in modulus size. The dispersion of run times is similar across different devices.

berry Pi devices are lower power, they represent a lower bound for hardware that may reasonably be expected to be used in practice outside of embedded applications.

We also executed performance tests on a consumer grade desktop PC. The machine used a quad-core 3.2 GHz Intel i5 processor, with 16 GB of available memory. We wished to confirm that the statistical properties remained constant over different hardware types. Additionally, this dataset provides a more pragmatic view of performance on commercial hardware.

Figure 6.1 demonstrates our first experiment.

This experiment shows how the run time of Gen, Solve, Encrypt and Decrypt is impacted by the time parameter t for a 2048 bit modulus when run on a desktop PC. The figure shows that as t increases, the run time for Solve also increases in a predictable linear manner. The linear variance in run time of Algorithm 16 Solve as t varies supports the RSW time-lock assumption in Definition 1. Furthermore, we see that Algorithm 15 Gen and Algorithms Encrypt and Decrypt remain consistently low, regardless of the size of t. This is expected as both algorithms reduce the parameter t by the group order using the Fermat-Euler Theorem 2. We also observe that Gen has minor variations in the run time when compared to Solve and Encrypt and Decrypt. This is a result of the randomised nature of Gen in comparison to the deterministic behaviour of Solve, Encrypt, and Decrypt.

For our next experiments we select $t = 5 \times 10^6$ to provide a total run time appropriate for repeat testing. We performed experimentation over four modulus sizes $m \in \{2048, 3072, 4096, 8192\}$ bit, selected to cover common modulus sizes in use. For each modulus size, we run 70 experiments, which allows us to estimate values with a 90% confidence interval with a 10% margin for error. The 8192 bit modulus is included as an edge case to demonstrate performance at the upper bound present in real world applications.

In Figure 6.2, we plot the spread of run times for the Gen algorithm. The primary metric of interest is the spread of the stochastic algorithm. For both the Pi and PC datasets, an increase in the modulus size increases the median run time. However, there is some overlap between modulus sizes, particularly between $m = 3072$ bit and $m = 4096$ bit. This discrepancy can be attributed to more efficient computation afforded when $\log_2 m \in \mathbb{N}$. In particular, the Miller-Rabin primality implementation can use a fast Fourier transform which is most efficient when dealing with powers of two [135]. The dispersion of the data points follows a similar pattern across both data sets, with an offset in median speed afforded by the relative difference in processor speed.

In Figure 6.3 we plot the run times of the Solve and Encrypt and Decrypt algorithms for both datasets against the modulus size. We use the run time means as a metric to eliminate variations caused by other processes on the machine, which we assume to be Gaussian. This leaves us with a more accurate indication of the run time of the deterministic algorithms. As with the Gen algorithm, we see similar increases in run time as a function of modulus size for both Solve and Encrypt and Decrypt. However, we note the large difference between the run times of Encrypt and Decrypt and Solve. Above each bar we plot the ratio of the run time of Encrypt and Decrypt to the run time of Solve. We see that, while there is a small increase in the ratio in relation to the modulus size, the difference between the two remains marked. Even at the edge case, when Solve runs in excess of four hours on the Pi when $m = 8192$ bit, Encrypt and Decrypt do not exceed 30 s. For most practical cases, Encrypt and Decrypt often results in sub-second evaluations, demonstrating practicality even in constrained environments. As Solve factors N, our TIDE construction is single-use for each auction. However, as we have seen in our experiments, this property is not an obstacle for practical use. Even in more computationally constrained environments such as the Pi, the Solve and Encrypt and Decrypt algorithms do not require an impractical time cost. Although we would recommend for a standard use case to use $m \leq 4096$ to keep the setup run time within an appropriate bound. This leaves the value for t as the primary parameter dictating the length of the delay. As we saw in Figure 6.1, the value for t can be set with reasonable

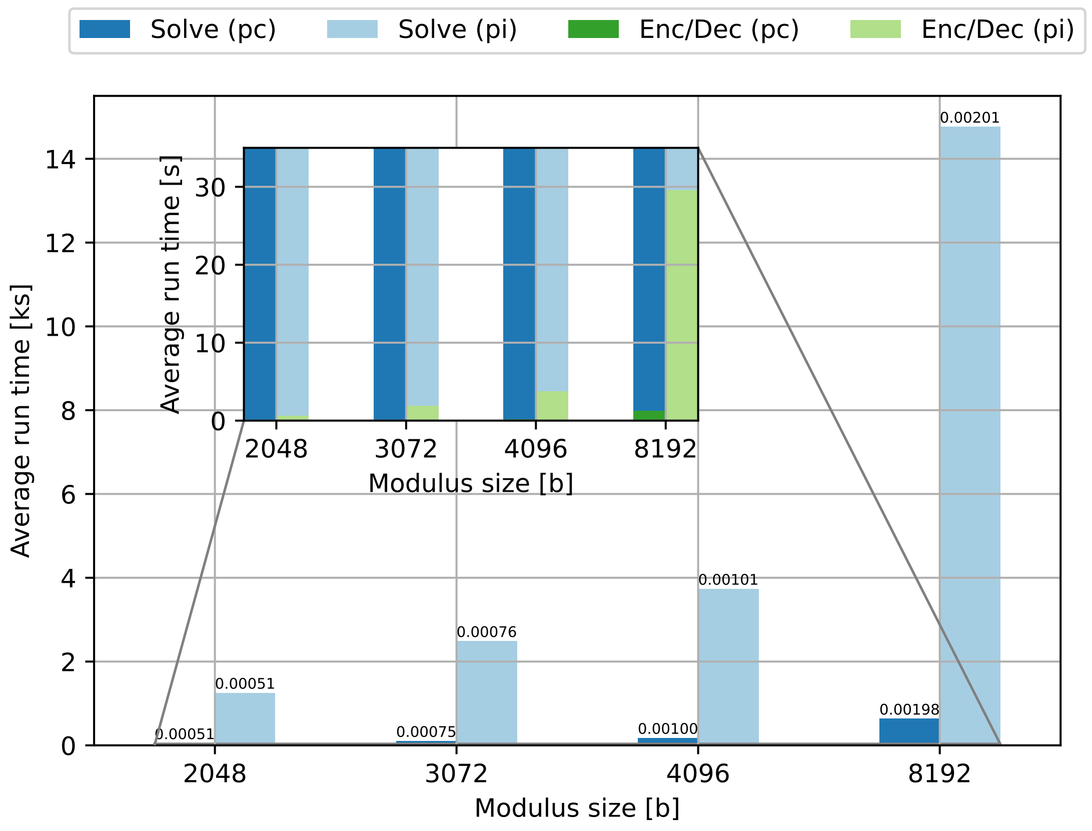

Figure 6.3: Encrypt and Decrypt take significantly less time to run than Solve across modulus sizes. The inset shows a zoomed in view of the bar chart which is necessary for the effect of Encrypt and Decrypt to be observed. Above each bar is the ratio Solve to Encrypt and Decrypt run time.

accuracy to introduce a desired delay for the target hardware.

6.5.2 RSA Decryption Optimisation

Various methods exist to optimise RSA calculations. One such method is the use of Montgomery multiplication [130]. Montgomery multiplication is a technique used to speed up modular multiplication, which is a fundamental operation in RSA decryption. The standard method for modular multiplication involves multiplying two numbers and then reducing the result modulo N. However, Montgomery multiplication modifies this process to reduce the number of modular reductions, making it more efficient.

Another method that can be used in conjunction with Montgomery multiplication to optimise RSA calculation is the use of the Chinese Remainder Theorem in the decryption process. The use of CRT in RSA decryption was first discussed by Quisquater and Couvreur [144]. The CRT RSA decryption process proceeds using the factors p and q directly.

This is particularly important in our TIDE construction as Solve recovers the factors p and q of the modulus N. In lines $3, 4, 5$ textbook RSA is used to perform the decryption process by using the decryption exponent d. This could be adapted to use the CRT-based decryption optimisation depicted by Quisquater and Couvreur [144].

6.5.3 Lossiness with the Encryption Exponent

We conclude our security discussion by addressing an important aspect in the context of our TIDE construction, namely, the lossiness with respect to the fixed encryption exponent. Lossiness, in this context, refers to the loss of information that occurs when using a function where the image is smaller than the domain, that is, the function is not injective.

To illustrate this concept, consider the example in RSA, where a 'lossy key' refers to a key pair (N, e) for which e divides $\phi(N)$. This results in information loss compared to a 'standard key' where e does not divide $\phi(N)$.

In the case where $e \mid \phi(N)$, where e is a lossy key, the mapping function from $x \to x^e$ is not a permutation. Instead $x \to x^e$ is the map e-to-1 on $\mathbb{Z}_N^*$. However, with a standard key, where e a non-lossy key, the mapping is one-to-one [158].

This consideration is particularly relevant in our TIDE implementation, where we employ a fixed value of $e = 65537$. The use of a lossy key has been demonstrated to weaken the security of RSA-based signature schemes like the Full Domain Hash (FDH) signature [98]. More critically, it has implications for repeated squaring when operating with Blum integers [156]. In the literature, extensive research has explored the challenge of distinguishing between cases where $\mathsf{gcd}(e, \phi(N)) = 1$ and the cases where e divides $\phi(N)$, especially when $e < N^{\frac{1}{4}}$. The latter scenario is known as the ϕ-hiding assumption [55, 156]. This consideration becomes particularly relevant in scenarios where N is generated through a distributed setup, as discussed in Section 4.3.3. However, due to the trusted setup of TIDE, we can incorporate a check to ensure that e does not divide $\phi(N) = (p-1)(q-1)$. This additional verification step safeguards against the selection of a lossy key. This is demonstrated in Algorithm 20.

Algorithm 20: GenNoLossy run on security parameter 1^κ and time parameter t to create the secret key sk, public key pk without lossiness and puzzle P.

```
   input : $1^\kappa, t, e \coloneqq 65537$
1  $p, q \coloneqq 2$
2  while $p = q \wedge e \nmid (p-1)(q-1)$ do
3      $p \coloneqq \mathsf{prime}(\frac{\kappa}{2})$
4      $q \coloneqq \mathsf{prime}(\frac{\kappa}{2})$
5  end
6  $N \coloneqq pq$
7  $\mathcal{J}_p(x), \mathcal{J}_q(x) \coloneqq 1$
8  while $\neg(\mathcal{J}_p(x) = 1 \wedge \mathcal{J}_q(x) \neq 1) \wedge \neg(\mathcal{J}_p(x) \neq 1 \wedge \mathcal{J}_q(x) = 1)$ do
9      $x \coloneqq \mathcal{U}(2, N)$
10     $\mathcal{J}_p(x) \coloneqq x^{\frac{p-1}{2}} \bmod p$
11     $\mathcal{J}_q(x) \coloneqq x^{\frac{q-1}{2}} \bmod q$
12 end
13 $x_0 \coloneqq x^2 \bmod N$
14 $\alpha_t \coloneqq x_0^{\frac{p+1}{4}^t \bmod p-1} \bmod p$
15 $\beta_t \coloneqq x_0^{\frac{q+1}{4}^t \bmod q-1} \bmod q$
16 $x_{-t} \coloneqq \alpha_t q(q^{-1} \bmod p) + \beta_t p(p^{-1} \bmod q) \bmod N$
17 $P \coloneqq (x, x_0, x_{-t})$
   output: $(\mathsf{sk}, \mathsf{pk}, P, t)$
```

6.6 Conclusion

In this work we introduced TIDE, a new TRE construction which seamlessly integrates the RSA cryptosystem with OAEP padding into a time-lock puzzle using classic number-theoretic concepts. TIDE challenges a solver to factor a special class of RSA modulus, known as a Blum integer. Parties may encrypt to this RSA modulus and any solver who factors the modulus may easily decrypt all encrypted messages. We demonstrated that this property makes TIDE well-suited to sealed-bid auctions: we compared TIDE to the most recent constructions for sealed-bid auctions, showing that TIDE has advantages both in terms of practicality and efficiency. We proved security of TIDE in the TRE framework introduced by Chvojka et al., and we implemented TIDE on both a Raspberry Pi and on a desktop PC, showing that it is indeed a practical construction.

Chapter 7

Outlook of Delay-Based Cryptography

In this chapter we examine the prospects and research directions for the timed-release encryption scheme with implicit authentication (TRE-IA) and the time delayed encryption (TIDE) scheme, presented in Part 2 of this thesis.

We first offer a convenient summary for reference comparing the TRE-IA and TIDE schemes. This comparison will help identify the similarities and differences between these schemes, demonstrating their respective technical details in the context of delay-based cryptography.

Next we provide the outlook for TRE-IA which involves software-based implementation and field research. Furthermore, we explore the integration of Post-Quantum Cryptography as a pressing concern. In the case of TIDE, we also consider the incorporation of Post-Quantum Cryptography as a pivotal area for future exploration. We finally provide an overview of our latest research topic which aims to use our TIDE construction as the basis for a public randomness beacon. These directions of further research seek to enhance the subject of delay-based cryptography.

7.1 Comparison of the TRE-IA and TIDE Schemes.

We will now provide a convenient summary which details the similarities and differences of the TRE-IA and TIDE schemes constructions. We will revisit the specific technical elements for which TRE-IA and TIDE were developed. TRE-IA was designed with a focus on providing secure tooling for whistleblowers, while TIDE was tailored to address the unique requirements of multiparty auctions. Table 7.1 summarises the constructions of our Part 2 schemes.

Table 7.1: TRE-IA vs. TIDE scheme feature comparison. Each row describes the key features of the TRE-IA and TIDE schemes. The challenge and the seed term of the time-lock puzzle are first identified. Next the time-lock puzzle calculation is shown. In the penultimate column the recovery of the decryption key d is described. For the TRE-IA scheme d is directly calculated, whereas with the TIDE scheme d is indirectly calculated by first recovering all square roots of x_t and then subsequently using the Congruence of Squares condition to recover the factors of N and then using EEA to recover the decryption key. Finally, we show that TIDE fixes the encryption exponent, while TRE-IA randomly selects the encryption exponent.

Scheme	**Challenge**	**Seed**	**Time-lock puzzle**	**Recover d**	**Fixed e**
TRE-IA	(x_0, x_t)	x_0	$d \coloneqq {x_0}^{2^{t-1}} \bmod N$	Directly	No
TIDE	(x, x_0, x_{-t})	x_{-t}	$x' \coloneqq (x_{-t})^{2^{t-1}} \bmod N$	Recover p, q w/ Definition 7 Recover d using EEA $d \coloneqq e^{-1} \bmod (p-1)(q-1)$	$e = 65537$

7.2 TRE-IA Outlook and Future Work

The outlook and future work for our timed-release encryption Scheme with implicit authentication (TRE-IA) for whistleblowers is to engage a concrete software-based implementation and conduct further field research and development. Our evaluation of TRE-IA has demonstrated its potential as a secure delay-based primitive with a user-friendly platform for whistleblowers. However, to ensure its robustness and address the relevant social science implications, careful considerations must be taken to ensure the security of any stakeholders engaging in our study. A great deal of consideration must be undertaken to assess the ethical considerations and security implications of our next research project.

By engaging in field testing with actual whistleblowers we can assess the practicality and usability of TRE-IA in real-world scenarios. By soliciting feedback from whistleblowers and incorporating their first-hand experiences, we can refine the design and ensure that TRE-IA integration with a tool such as SecureDrop meets their specific needs and requirements.

Furthermore, conducting usability studies involving participants from diverse backgrounds could enhance the user experience of TRE-IA. By observing how users interact with the system and collecting their feedback, we can identify areas for improvement, simplify the user interface, and ensure that TRE-IA is accessible to a wide range of potential and diverse whistleblowers.

Of utmost important beyond the technical aspects of a TRE-IA implementation, it will be imperative to consider the ethical implications and social consequences of deploying such potentially disruptive tooling. Research in social sciences can help us understand the impact of TRE-IA on whistleblowing dynamics, power structures, and legal considerations. Engaging multidisciplinary teams comprising experts in ethics, sociology, law, and psychology will provide insights to examine these complex issues and ensure the responsible deployment of TRE-IA.

7.2.1 Post-Quantum Cryptography and TRE-IA

The emergence of Post-Quantum Cryptography (PQC) features as a pressing concern. Briefly, PQC devises cryptographic problems that are resistant to quantum computing threats, unlike traditional cryptographic methods that rely on the hardness of problems like integer factorisation or the discrete logarithm problem. PQC leverages mathematical constructs such as lattice-based cryptography, code-based cryptography, and multivariate polynomial cryptography, among others [147, 146, 148, 59]. These approaches introduce problems that are currently believed to be difficult for quantum computers to solve efficiently.

For example, lattice-based cryptography involves the use of mathematical lattices, which are grids of points in multidimensional space. Finding short vectors in these lattices is considered computationally challenging and forms the basis of cryptographic algorithms. Similarly, code-based cryptography relies on the complexity of decoding random linear codes, and multivariate polynomial cryptography involves solving systems of multivariate polynomial equations, which become increasingly difficult to solve as the number of variables grows. These alternative mathematical problems offer a promising avenue for securing digital communication in a post-quantum era.

While TRE-IA, as presented in this thesis, relies on the presumed hardness of integer factorisation, a core problem underpinning RSA-based cryptography, the advent of quantum computing threatens the security of such classical cryptographic primitives. To ensure the continued relevance and resilience of TRE-IA in a quantum-powered computational model, future research should earnestly explore alternatives to conventional cryptographic components.

This pursuit should encompass the replacement of both Rivest-Shamir-Wagner (RSW) time-lock puzzles and the RSA Public-Key Encryption (PKE) system, both dependent on the hardness of integer factorisation. Embracing PQC solutions is paramount to maintaining the security of digital systems and safeguarding the privacy of whistleblowers. Therefore, the incorporation of PQC into TRE-IA represents a significant avenue for future research.

7.3 TIDE Outlook and Future Work

The future work for TIDE (TIme Delayed Encryption) holds promising prospects in the real-world practical implementation of this cryptographic scheme. A significant next step involves the development of a concrete software-based implementation of TIDE, paving the way for real-world testing and deployment. A primary focus should be on the practical application of TIDE in scenarios such as sealed-bid auctions, where its unique capabilities can be tested. This entails engaging in field research and development to assess the usability, security, and effectiveness of TIDE in authentic, sealed-bid auction environments. By soliciting feedback and insights from users participating in sealed-bid auctions, we can refine the design and ensure that TIDE meets their specific needs and requirements. This practical implementation-oriented research approach will enhance our understanding of how TIDE can be used in real-world applications.

7.3.1 Post-Quantum Cryptography and TIDE

Post-Quantum Cryptography also represents a concern for the security and longevity of delay-based cryptographic systems like TIDE. While TIDE, as presented in this chapter, relies on the presumed hardness of integer factorisation, a core problem underpinning RSA-based cryptography and RSW-based time-lock puzzles, the recent advances of quantum computing threaten the security of such classical cryptographic primitives.

To ensure the continued relevance and resilience of TIDE in a quantum-powered computational model, future research should earnestly explore alternatives to conventional cryptographic components. Similar to TRE-IA, research in this area should consider alternatives of both RSW-based time-lock puzzle constructions and the RSA PKE cryptosystem. Considering PQC solutions is essential to maintaining the security of delay-based cryptography and safeguarding the privacy of stakeholders which may rely on TIDE in a sealed-bid auction. Therefore, the incorporation of PQC into TIDE represents a significant subject for future research, one that aligns seamlessly with the goals in our TRE-IA construction.

Addressing these areas of future research will enhance the TIDE scheme and also contribute to the broader area of research and development regarding the topic of delay-based cryptography.

7.3.2 Randomness Beacon from TIDE

In our most recent research we are investigating the use of our TIDE construction to establish a randomness beacon [48] based on a continuous verifiable delay function (cVDF) [74].

Our current TIDE-based randomness beacon proposal has also been done in collaboration with Liam Medley and is currently in submission. All work is being completed under the supervision of Elizabeth Anne Quaglia.

A high-entropy source of public randomness is a necessary component of many cryptographic protocols, including secret sharing and key distribution [67, 46]. Since 2011 NIST have been running a competition to build a trusted randomness beacon, with the objective of promoting the availability of trusted public randomness as a public utility [100]. A randomness beacon allows a group of parties to use some shared randomness in a protocol, each with the guarantee that none of the other parties has any prior knowledge of the output. Common applications include running a lottery and random sampling. Random sampling can be used for selecting patients in clinical trials or selecting officials in audits, for instance.

A modern approach to building a randomness beacon utilises verifiable delay functions (VDFs) [48]. Whilst VDFs can also be used to construct consensus protocols and in timestamping, their flagship application is indeed to provide a publicly verifiable randomness beacon [63, 106].

These functions, initially introduced by Boneh et al. in 2018 have become a cornerstone in modern cryptographic research. On a high level, a VDF samples a pseudo-random input, and uses the output of the delay function to extract randomness. A VDF is characterised by its ability to impose a delay on computation through an iterated sequential function, ensuring that each input produces a unique output that can be efficiently and publicly verified.

Following the work by Boneh et al., several VDF candidates emerged. Notably, Wesolowski [166] and Pietrzak [141] independently proposed VDFs based on the RSA

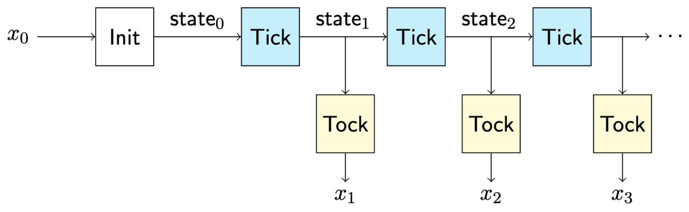

Figure 7.1: The Tick/Tock paradigm as illustrated by Ephraim et al. in [74], where each x_i represents a pulse of randomness.

group modulo N, relying on repeated squaring and reduction. Both designs are RSW TLP-based and both involve interactive protocols in which solvers prove to verifiers that they have correctly computed the solution.

In their EUROCRYPT 2019 paper, Ephraim et al. introduced the concept of a continuous verifiable delay function based on the VDF candidate from Pietrzak [74]. Unlike traditional VDFs, the cVDF model introduced by Ephraim et al. incorporates the concept of 'states', representing intermediate points within the computation that can be independently verified. This state-based approach unlocks two critical applications not feasible with standard VDFs. First, at any state, the party performing the computation can transfer it to another party, who can efficiently verify the state and continue the computation. Second, by verifying each state in the process, trusted public randomness can be extracted at regular intervals, effectively creating an efficient source of randomness that periodically emits values, aligning well with the concept of a randomness beacon (RB).

Importantly, Ephraim et al. demonstrated that a secure cVDF, meeting criteria such as adaptively soundness, sequentiality, and correctness, serves as a foundational component for constructing a secure randomness beacon. They implement their randomness beacon using the Tick/Tock paradigm, as illustrated in Figure 7.1.

In the context of such a beacon, the setup procedure generates an initial state, denoted as state_0, as shown in Figure 7.1 where Ephraim et al. refer to the relevant algorithm as Init taking input x_0. Subsequently, two algorithms run concurrently on every state: the Tick algorithm, which takes a state state_i and produces the next state state_{i+1}, and the Tock algorithm, which operates on state_i to yield a pulse of randomness, represented as x_i in the diagram. Verification procedures are then applied to both the Tick and Tock computations, demonstrating the correct computation of the state and the accurate derivation of randomness from the state.

This approach to constructing a randomness beacon, as introduced by Ephraim et al., presents opportunities for improvement. The definitions proposed in [74] rely on a very specific computational model, incorporating numerous non-standard parameters and nested definitions. This departure from the conventional delay-based literature can make these definitions challenging to adopt, necessitating a comprehensive understanding of this unique model. Additionally, the cVDF construction outlined in [74], along with its associated randomness beacon, exhibits a verification time that grows $O(\log t)$ with the time parameter, resulting in inefficient scalability as time progresses.

The observations regarding the opportunities for improvement in the randomness beacon construction proposed by Ephraim et al. have motivated our interest in further research in this area. In the following section, we will outline our research goals and the direction we intend to take in addressing these challenges and enhancing the efficiency and scalability of randomness beacons within delay-based cryptographic systems.

7.3.3 TIDE Randomness Beacon Research Objectives

Minimising dependence in central stakeholders is key to the implementation and adoption of verifiable delay functions and randomness beacons, as the security and reliability of these cryptographic primitives focus on reducing the level of reliance placed in the setup process. Balancing the need for efficient computation with the necessity of minimising reliance on a centralised stakeholder in the setup remains an ongoing challenge in the field of delay-based cryptography. The issue of reliance on a centralised stakeholder in RSW-based VDFs (such as that used in Ephraim et al.) remains unsolved, and while the use of MPC ceremonies is becoming increasingly practical, it presents its own set of challenges, including the risk of denial of service [114]. Furthermore, all RSW TLP-based VDFs require reliance on a centralised stakeholder in the generation of the modulus [141, 166, 74].

The value of public randomness is important for use cases such as unbiased statistical sampling and lotteries. Therefore, it is important in these use cases to consider the reduction of centralised dependency required in the setup process. In our next pursuit of developing a randomness beacon, we are committed to minimising reliance on any single source of randomness. By refining the core concepts of TIDE and its application to a randomness beacon, we are actively contributing to the advancement of dependable and transparent randomness generation mechanisms which form a crucial component in the space of delay-based cryptography. While this work is still in its early phases, we have established the fundamental research goals that will guide our forthcoming exploration in the realm of delay-based cryptography.

Our research objectives, concerning the utilisation of TIDE in establishing a randomness beacon, are outlined as follows:

1. Formulate an enhanced definition of a cVDF.
2. Demonstrate that the conformity of a cVDF to our definition implies conformity to a standard VDF.
3. Develop a cVDF utilising our TIDE primitive.
4. Construct a randomness beacon incorporating our TIDE-based cVDF utilising the Tick/Tock paradigm.
5. Validate that our randomness beacon satisfies all pertinent security and design criteria.

Our first research goal is to consider improvements to the Ephraim et al., cVDF definition to standardise it against the original VDF definition given by Boneh et al. [48]. The definitions given by the latter are heavily parameterised, which has the drawback of restricting the user to a specific computational model. We aim to create definition for a primitive to be as generic as possible, which would allow users who work within various computational models to be able to use these definitions. Our definitions aim to have fewer parameters and will be significantly closer to the generic VDF definitions.

Our second research objective is to formally establish that, by simplifying the cVDF parameters and introducing a single 'continuity parameter', our new cVDF Definition aligns with the conventional VDF Definition.

Our third and fourth research objectives will consist of leveraging the use of our TIDE primitive to first create a cVDF and subsequently create a secure randomness beacon.

From a centralised dependency perspective, the entity responsible for executing the generation of the public parameters and the RSW-based time-lock puzzle holds a significant degree of power. Not only are they privy to the factors of the modulus N, but without a novel construction, they could also possess the ability to precompute every random element ahead of the intended time of release. Therefore, we are actively exploring strategies to minimise this centralised dependency, aiming to limit the amount of reliance placed on the party with knowledge of the factors of the modulus N.

Once we refine the creation of our cVDF through the TIDE construction, the subsequent stages of construction should proceed more smoothly. Lastly, any secure cVDF can yield a randomness beacon in the Tick/Tock paradigm [74]. Therefore, upon constructing our cVDF, we plan to adapt for the construction of a randomness beacon. Finally, once we are able to construct a randomness beacon from a cVDF predicated on the use of our TIDE construction we will then proceed to address our fifth research objective – which is to ensure that our RB satisfies all security and design criteria. We believe this is a valuable piece of research in the area of delay-based cryptography, and we are currently working on this as part of the outlook and future work related to TIDE.

7.4 Conclusion

In the final part of our thesis we provided two practical implementations of delay-based cryptography, exploring their potential applications for whistleblowing encryption and sealed-bid auctions. Through the examination of RSW time-lock puzzles and RSA-OAEP based timed-release encryption schemes, we have creatively combined various cryptographic primitives to create our unique constructions to strategically tackle the research gaps identified at the inception of our thesis. Our first construction TRE-IA introduced the property of implicit authentication, which provides assurance that only the party holding the encryption key can encrypt a document of their choosing and produce a valid ciphertext which correctly decrypts to a meaningful plaintext. Our second construction TIDE was created to address practical challenges inherent in current timed-release encryption schemes used in the context of sealed-bid auctions. Looking ahead, the field of delay-based cryptography still has many areas for further advancements, particularly considering emerging challenges such as post-quantum cryptography. Finally, we presented our ongoing efforts to utilise our TIDE construction for creating a randomness beacon. This work in progress exemplifies the innovation that could be expanded using this primitive.

Chapter 8

Bibliography

[1] Edward Snowden's Motive Revealed: He Can 'Sleep at Night', 2014. https://www.nbcnews.com/feature/edward-snowden-interview/edward-snowdens-motive-revealed-he-can-sleep-night-n116851.

[2] GDPR - Personal Data. Technical report, European Parliament and Council of the European Union, 2016. https://gdpr-info.eu/issues/personal-data.

[3] Evaluate, possibly revise, and then implement ideas for TLS certificate normalization, 2017. https://trac.torproject.org/projects/tor/ticket/7145.

[4] Bitcoin Developer Documentation: Technical Glossary - Fork, 2018. https://bitcoin.org/en/glossary/fork.

[5] Electroneum README.md - Using Tor, 2018. https://github.com/electroneum/electroneum/blob/master/README.md.

[6] Hard Fork News, 2018. https://cointelegraph.com/tags/hard-fork.

[7] ICO Launch Calendar, 2018. https://coinlauncher.io.

[8] Monero README.md - Using Tor, 2018. https://github.com/monero-project/monero/blob/master/README.md.

[9] Regulation of Cryptocurrency Around the World . Technical report, Law Library of Congress - Global Legal Research Center, 2018. http://www.loc.gov/law/help/cryptocurrency/cryptocurrency-world-survey.pdf.

[10] Top 100 Cryptocurrencies by Market Capitalization, 2018. https://coinmarketcap.com.

[11] U.S. Securities and Exchange Commission - Initial Coin Offerings (ICOs), 2018. https://www.sec.gov/ICO.

[12] Welcome to RIPE Atlas, 2018. https://atlas.ripe.net.

[13] Donate to WikiLeaks, 2019. https://shop.wikileaks.org/donate.

[14] IRC Networks and Servers, 2019. https://www.mirc.co.uk/servers.html.

[15] OCCRP Donate, 2019. https://www.occrp.org/en/donate.

[16] The Tor Project, 2019. https://www.torproject.org.

[17] SecureDrop Whistleblower Submission System, 2021. https://securedrop.org.

[18] Signal Messaging, 2021. https://signal.org/en.

[19] Download Bitcoin Core, 2022. https://bitcoin.org/en/download.

[20] H. Abdi. Coefficient of Variation. In *Encyclopedia of research design*, 2010.

[21] G. Aceto and A. Pescapé. Internet Censorship detection: A survey. *Computer Networks*, 83:381–421, 2015.

[22] D. Adrian, Z. Durumeric, G. Singh, and J.A. Halderman. Zippier ZMap: Internet-wide scanning at 10 Gbps. In *8th USENIX Workshop on Offensive Technologies (WOOT 14)*, San Diego, CA, USA, 2014.

[23] A. Alexandre. New York State Regulators Approve New Power Rate Structure for Crypto Miners, 2018. https://tinyurl.com/uh43bh66.

[24] A. Alexandre. Brazilian Police Arrest Suspect for Money Laundering With Bitcoin, 2019. https://cointelegraph.com/news/brazilian-police-arrest-suspect-for-money-laundering-with-bitcoin.

[25] Anonymous. The Collateral Damage of Internet Censorship by DNS Injection. *ACM SIGCOMM Computer Communication Review*, 42(3):21–27, 2012.

[26] A. Antonopoulos. *Mastering Bitcoin: Unlocking Digital Cryptocurrencies*. O'Reilly Media, Sebastopol, CA, USA, 2 edition, 2017.

[27] M. Ashton. Worldwide broadband speed league 2018, 2018. https://www.cable.co.uk/broadband/speed/worldwide-speed-league.

[28] L. Ausubel. A generalized Vickrey auction. *Econometrica*, 1999.

[29] E. Bach and J. Shallit. *Algorithmic number Theory, Vol. 1: Efficient Algorithms*. MIT Press, 1996.

[30] B. Barak, O. Goldreich, R. Impagliazzo, S. Rudich, A. Sahai, S. Vadhan, and K. Yang. On the (im) possibility of obfuscating programs. In *Annual international cryptology conference*. Springer, 2001.

[31] E. Barker, L. Chen, A. Roginsky, A. Vassilev, R. Davis, and S. Simon. NIST Special Publication 800-56B Revision 2. Recommendation for Pair-Wise Key Establishment Using Integer Factorization Cryptography, 2019.

[32] E. Barker, L. Chen, A. Roginsky, A. Vassilev, R. Davis, and S. Simon. SP 800-56B Rev. 2, Recommendation for Pair-Wise Key-Establishment Using Integer Factorization Cryptography. *ITL Computer Security Resource Center*, 2019.

[33] S. Bates, J. Bowers, S. Greenstein, J. Weinstock, J. Sittrain, and Y. Xu. Evidence of Decreasing Internet Entropy: The Lack of Redundancy in DNS Resolution by Major Websites and Services. *Working Paper 24317, National Bureau of Economic Research*, 2018. https://www.hbs.edu/faculty/Pages/item.aspx?num=53830.

[34] A. Baydakova. Russian Opposition Leader Raises Three Million in Bitcoin, 2019. https://www.coindesk.com/russian-opposition-leader-raises-3-million-in-bitcoin-donations.

[35] M. Beedham. An American woman has been funding ISIS with Bitcoin, 2018. https://thenextweb.com/hardfork/2018/11/27/american-funding-isis-bitcoin.

[36] M. Bellare and C. Namprempre. Authenticated encryption: Relations among notions and analysis of the generic composition paradigm. In *Advances in Cryptology—ASIACRYPT 2000: 6th International Conference on the Theory and Application of Cryptology and Information Security Kyoto, Japan, December 3–7, 2000 Proceedings 6*, pages 531–545. Springer, 2000.

[37] M. Bellare and P. Rogaway. Optimal Asymmetric Encryption. In *Advances in Cryptology EUROCRYPT '94*, 1994.

[38] E. Ben-Sasson, A. Chiesa, C. Garman, M. Green, I. Miers, E. Tromer, and M. Virza. Zerocash: Decentralized Anonymous Payments from Bitcoin(extended version), 2014. http://zerocash-project.org/media/pdf/zerocash-extended-20140518.pdf.

[39] P. Berglez and A. Gearing. The Panama and Paradise Papers. The rise of a global fourth estate. *International Journal of Communication*, 2018.

[40] F. Berti, F. Koeune, O. Pereira, T. Peters, and F. Standaert. Ciphertext integrity with misuse and leakage: definition and efficient constructions with symmetric primitives. In *Proceedings of the 2018 on Asia Conference on Computer and Communications Security*, pages 37–50, 2018.

[41] A. Biryukov and I. Pustogarov. Bitcoin over Tor isn't a Good Idea. In *IEEE Symposium on Security and Privacy*, pages 112–134, San Jose, California, 2015.

[42] N. Bitansky, S. Goldwasser, A. Jain, O. Paneth, V. Vaikuntanathan, and B. Waters. Time-lock puzzles from randomized encodings. In *Proceedings of the 2016 ACM Conference on Innovations in Theoretical Computer Science*, 2016.

[43] E. Blass and F. Kerschbaum. Borealis: Building block for sealed bid auctions on blockchains. In *Proceedings of the 15th ACM Asia Conference on Computer and Communications Security*, 2020.

[44] L. Blum, M. Blum, and M. Shub. A Simple Unpredictable Pseudo-Random number Generator. In *Journal on Computing*, 1986.

[45] A. Blume and P. Heidhues. All equilibria of the Vickrey auction. *Journal of economic Theory*, 2004.

[46] C. Blundo, A. De Santis, and U. Vaccaro. Randomness in distribution protocols. *Information and Computation*, 1996.

[47] P. Bogetoft, D. Christensen, I. Damgård, M. Geisler, T. Jakobsen, M. Krøigaard, J. Nielsen, J. Nielsen, K. Nielsen, J. Pagter, et al. Secure multiparty computation goes live. In *International Conference on Financial Cryptography and Data Security*. Springer, 2009.

[48] D. Boneh, J. Bonneau, B. Bünz, and B. Fisch. Verifiable Delay Functions. In *Advances in Cryptology - CRYPTO 2018 - 38th Annual International Cryptology Conference*, 2018.

[49] D. Boneh and M. Naor. Timed commitments. In *Annual international cryptology conference*. Springer, 2000.

[50] Dan Boneh and Matthew Franklin. Identity-based encryption from the Weil pairing. In *Advances in Cryptology—CRYPTO 2001*, pages 213–229. Springer, 2001.

[51] F. Brandt. Auctions. In *Handbook of Financial Cryptography and Security*. Chapman and Hall/CRC, 2010.

[52] G. Bruneau and R. Wanner. DNS Sinkhole, 2010. https://www.sans.org/reading-room/whitepapers/dns/dns-sinkhole-33523.

[53] J. Burdges and J. DeFeo. Delay Encryption. In *40th Annual International Conference on the Theory and Applications of Cryptographic Techniques. EUROCRYPT 2021*, 2021.

[54] Vitalik Buterin. A Next Generation Smart Contract & Decentralized Application Platform. *White paper*, 2013.

[55] C. Cachin, S. Micali, and M. Stadler. Computationally private information retrieval with polylogarithmic communication. In *Advances in Cryptology – EUROCRYPT'99 International Conference on the Theory and Application of Cryptographic Techniques*, pages 302–414, 1999.

[56] R. Carmichael. Note on a new number theory function. In *Bulletin of the American Mathematical Society*, 1910.

[57] J. Cathalo, B. Libert, and J. Quisquater. Efficient and non-interactive timed-release encryption. In *International Conference on Information and Communications Security*. Springer, 2005.

[58] Krishnendu Chatterjee, Amir K. Goharshady, and Arash Pourdamghani. Hybrid Mining: Exploiting Blockchain's Computational Power for Distributed Problem Solving. In *Proceedings of the ACM Symposium on Applied Computing*, 2019.

[59] L. Chen, S. Jordan, Y. Liu, D. Moody, R. Peralta, R. Perlner, and D. Smith-Tone. Report on Post-Quantum Cryptography (NISTIR8105), 2016.

[60] M. Chen, Jack Doerner, Y. Kondi, E.Lee, S. Rosefield, A. Shelat, and R. Cohen. Multiparty generation of an RSA modulus. *Journal of Cryptology*, 2022.

[61] M. Chen, C. Hazay, Y. Ishai, Y. Kashnikov, D. Micciancio, T. Riviere, A. Shelat, M. Venkitasubramaniam, and R. Wang. Diogenes: lightweight scalable RSA modulus generation with a dishonest majority. In *2021 IEEE Symposium on Security and Privacy (SP)*, 2021.

[62] P. Chvojka, T. Jager, D. Slamanig, and C. Striecks. Versatile and sustainable timed-release encryption and sequential time-lock puzzles. In *European Symposium on Research in Computer Security*. Springer, 2021.

[63] B. Cohen and K. Pietrzak. The chia network blockchain, 2019.

[64] J. Cook. RSA encryption exponents are mostly all the same. https://www.johndcook.com/blog/2018/12/12/rsa-exponent, 2018.

[65] J. Cook. Computing Legendre and Jacobi symbols, Algorithm for computing Jacobi symbols. https://www.johndcook.com/blog/2019/02/12/computing-jacobi-symbols, 2019.

[66] T. Cormen, C. Leiserson, R. Rivest, and C. Stein. *Introduction to Algorithms*. MIT Press, 2009.

[67] H. Corrigan-Gibbs, W. Mu, D. Boneh, and B. Ford. Ensuring high-quality randomness in cryptographic key generation. In *Proceedings of the 2013 ACM SIGSAC conference on Computer & communications security*, 2013.

[68] R. Crandall and C. Pomerance. *Prime numbers: A Computational Perspective*. Springer-Verlag, 2005.

[69] A. de Vries. Bitcoin's Growing Energy Problem, 2018. https://www.cell.com/joule/pdf/S2542-4351(18)30177-6.pdf.

[70] E. Van der Sar. Pirate Bay Moves to The Cloud, Becomes Raid-Proof, 2012. https://torrentfreak.com/pirate-bay-moves-to-the-cloud-becomes-raid-proof-121017.

[71] E. Van der Sar. The Pirate Bay Turns 15 Years Old, 2018. https://torrentfreak.com/the-pirate-bay-turns-15-years-old-180810.

[72] J.A.D. Donet, C. Pérez-Solà, and J. Herrera-Joancomartí. The Bitcoin P2P Network. In *Bitcoin '14: Proceedings of the 1st Workshopon Bitcoin Research*, Barbados, 2014.

[73] E. Duffield and D. Diaz. Dash: A Payments-Focused Cryptocurrency, 2015. https://github.com/dashpay/dash/wiki/Whitepaper.

[74] N. Ephraim, C. Freitag, I. Komardogski, and R. Pass. Continuous Verifiable Delay Functions. In *Advances in Cryptology - EUROCRYPT 2020 - 39th Annual International Conference on the Theory and Applications of Cryptographic Techniques*, 2020.

[75] L. De Feo, S. Masson, C. Petit, and A. Sanso. Verifiable Delay Functions from Supersingular Isogenies and Pairings. In *Advances in Cryptology – ASIACRYPT 2019 – 25th Annual Conference*, 2019.

[76] J. Fraleigh. *A First Course in Abstract Algebra*. Pearson, Harlow, UK, 2014.

[77] J. Frankenfield. Off-Chain Transactions, 2018. https://www.investopedia.com/terms/o/offchain-transactions-cryptocurrency.asp.

[78] T. Frederiksen, Y. Lindell, V. Osheter, and B. Pinkas. Fast distributed RSA key generation for semi-honest and malicious adversaries. In *Annual International Cryptology Conference*. Springer, 2018.

[79] C. Freitag, I. Komargodski, R. Pass, and N. Sirkin. Non-malleable Time-Lock Puzzles and Applications. In *Theory of Cryptography Conference*. Springer, 2021.

[80] J. Friedlander, C. Pomerance, and I. Shparlinski. Period of the power generator and small values of Carmichael's function. In *American Mathematical Society*. Mathematics of Computation 70, 2000.

[81] E. Fujisaki, T. Okamoto, D. Pointcheval, and J. Stern. RSA-OAEP is secure under the RSA assumption. In *Annual International Cryptology Conference*. Springer, 2001.

[82] H. Galal and A. Youssef. Verifiable sealed-bid auction on the ethereum blockchain. In *International Conference on Financial Cryptography and Data Security*. Springer, 2018.

[83] J. Garside. Panama Papers: inside the Guardian's investigation into offshore secrets, 2016. https://www.theguardian.com/news/2016/apr/16/panama-papers-inside-the-guardians-investigation-into-offshore-secrets.

[84] C. Gauss. *Disquisitiones Arithmeticae.* Yale University Press, 2009.

[85] C. GauthierDickey and C. Grothoff. Bootstrapping of Peer-to-Peer Networks. In *Proceedings of the International Symposium of Applications and the Internet (SAINT'08)*, pages 205–208, Turku, Finland, 2008.

[86] Gearlog. LimeWire, Napster, The Pirate Bay: A Brief History of File Sharing, 2010. https://www.geek.com/gadgets/limewire-napster-the-pirate-bay-a-brief-history-of-file-sharing-1359473.

[87] T. Goldenberg. Watch Out Crypto Exchanges, Decentralization Is Coming, 2018. https://www.coindesk.com/future-crypto-exchanges-decentralization-coming.

[88] R.L. Goodstein. *Boolean Algebra.* Dover Publications, 2007.

[89] W. Gordon. What Are Magnet Links, and How Do I Use Them to Download Torrents?, 2012. https://lifehacker.com/5875899/what-are-magnet-links-and-how-do-i-use-them-to-download-torrents.

[90] B. Greschbach, T.Pulls, L.M. Roberts, P. Winter, and N. Feamster. The Effect of DNS on Tor's Anonymity, 2016. https://arxiv.org/pdf/1609.08187.pdf.

[91] F. Griffin and I. Shparlinski. On the linear complexity profile of the power generator. In *IEEE Transactions on Information Theory*, 2000.

[92] J.M. Griffin and A. Shams. Is Bitcoin Really Un-Tethered?, 2018. https://papers.ssrn.com.

[93] G. Hardy and E. Wright. *An introduction to the theory of numbers.* Oxford university press, 1979.

[94] C. Hazay, G. Mikkelsen, R. Rabin, T. Toft, and A. Nicolosi. Efficient RSA key generation and threshold paillier in the two-party setting. *Journal of Cryptology*, 2019.

[95] E. Heilman, A. Kendler, A. Zohar, and S. Goldberg. Eclipse Attacks on Bitcoins Peer-to-Peer Network. In *24th USENIX Security Symposium (USENIX Security 15)*, pages 129–144, Washington, D.C, 2015.

[96] A. Jain, H. Lin, and A. Sahai. Indistinguishability obfuscation from well-founded assumptions. In *Proceedings of the 53rd Annual ACM SIGACT Symposium on Theory of Computing*, 2021.

[97] A. Juels and M. Szydlo. A two-server, sealed-bid auction protocol. In *International conference on financial cryptography*. Springer, 2002.

[98] S. Kakvi and E. Kiltz. Optimal security proofs for full domain hash, revisited. In *Advances in Cryptology – EUROCRYPT'12 International Conference on the Theory and Application of Cryptographic Techniques*, pages 537–553, 2012.

[99] J. Katz and Y. Lindell. *Introduction to Modern Cryptography, Second Edition*. CRC Press, 2014.

[100] J. Kelsey, L. Brandão, R. Peralta, and H. Booth. A Reference for Randomness Beacons. Format and Protocol Version 2, 2016. https://nvlpubs.nist.gov/nistpubs/ir/2019/NIST.IR.8213-draft.pdf.

[101] J. Kennedy. The Pirate Bay: Sailing away from torrent files next month, 2012. http://www.extremetech.com/computing/113767-pirate-bay-sailing-away-from-torrent-files-next-month.

[102] E. Kim. Venezuela Bitcoin Trading Record Underscores Fiat Currency Hyperinflation, 2019. https://tinyurl.com/2uurud5f.

[103] M. Knoll, M. Helling, A. Wacker, S. Holzapfel, and T. Weis. Bootstrapping Peer-to-Peer Systems using IRC. In *18th IEEE International Workshops Enabling Technologies: Infrastructures for Collaborative Enterprises*, pages 122–127, Groningen, Netherlands, 2009.

[104] M. Knoll, A. Wacker, G. Schiele, and T. Weis. Bootstrapping in Peer-to-Peer Systems. In *14th IEEE International Conference on Parallel and Distributed Systems*, pages 271–278, Melbourne, Australia, 2008.

[105] R. Koch. Here are all the countries where the government is trying to ban VPNs, 2018. https://protonvpn.com/blog/are-vpns-illegal.

[106] E. Landerreche, M. Stevens, and C. Schaffner. Non-interactive cryptographic timestamping based on verifiable delay functions. In *International Conference on Financial Cryptography and Data Security*. Springer, 2020.

[107] M. Leising. The Ether Thief, 2017. https://www.bloomberg.com/features/2017-the-ether-thief.

[108] A. Lenstra and I. Shparlinski. Selective forgery of RSA signatures with fixed-pattern padding. In *Public Key Cryptography: 5th International Workshop on Practice and Theory in Public Key Cryptosystems*, 2002.

[109] M. Liedtke and J. Mattise. Leaked 'Pandora Papers' expose how billionaires and corrupt leaders hide wealth. *Guardian (Sydney)*, 2021.

[110] C. Liu and P. Albitz. *DNS and BIND*. O'Reilly Media, Sebastopol, CA, USA, 5 edition, 2006.

[111] J. Liu, F. Garcia, and M. Ryan. Time-release protocol from bitcoin and witness encryption for sat. *Korean Circulation Journal*, 2015.

[112] A. Loe, L. Medley, C. O'Connell, and E. Quaglia. TIDE: A Novel Approach to Constructing Timed-release Encryption. In *Proceedings of the 2022 ACISP: Australasian Conference on Information Security and Privacy*, pages 244–264, Wollongong, Australia, 2022.

[113] A. Loe, L. Medley, C. O'Connell, and E. Quaglia. Applications of Timed-Release Encryption with Implicit Authentication. In *Proceedings of the 14th International Conference on Cryptology in Africa, AFRICACRYPT 2023*, pages 490–515, Sousse, Tunisia, 2023.

[114] A. Loe, L. Medley, and E. Quaglia. SoK: Delay-Based Cryptography. In *Proceedings of the 36th IEEE Computer Security Foundations Symposium 2023*, pages 426–440, Dubrovnik, Croatia, 2023.

[115] A. Loe and E. Quaglia. Conquering Generals: an NP-Hard Proof of Useful Work. In *Proceedings of the 1st Workshop on Cryptocurrencies and Blockchains for Distributed Systems*, pages 54 – 59, 2018.

[116] A. Loe and E. Quaglia. You Shall Not Join: A Measurement Study of Cryptocurrency Peer-to-Peer Bootstrapping Techniques. In *Proceedings of the 2019 ACM*

SIGSAC Conference on Computer and Communications Security, pages 2231–2247, London, United Kingdom, 2019.

[117] E.K. Lua, J. Crowcroft, M. Pias, R. Sharma, and S. Lim. A Survey and Comparison of Peer-to-Peer Overlay Network Schemes. *IEEE Communications Surveys and Tutorials*, 7(2), 2005.

[118] S. Lui. How To Bypass ISP Blocking Of The Pirate Bay And Other Torrent Sites For Free, 2018. https://tinyurl.com/uh43bh66.

[119] A. Mairh, D. Barik, K. Verman, and D. Jenna. Honeypot in network security: a survey. In *ACM International Conference on Communication, Computing and Security*, pages 600–605, Odisha, India, 2011.

[120] G. Malavolta and S. Thyagarajan. Homomorphic time-lock puzzles and applications. In *Annual International Cryptology Conference*. Springer, 2019.

[121] W. Mao. Timed-release cryptography. In *International Workshop on Selected Areas in Cryptography*. Springer, 2001.

[122] S. Matic, C. Troncoso, and J. Caballero. Dissecting Tor Bridges: a Security Evaluation of Their Private and Public Infrastructures. In *Network and Distributed Systems Security Symposium. The Internet Society*, pages 1 – 15, San Diego, CA, USA, 2017.

[123] T. May. Timed-release crypto. 1993. http://www.hks.net.cpunks/cpunks-0/1560.html.

[124] S. Meredith. Bitcoin trading in crisis-stricken Venezuela has just hit an all-time high, 2019. https://www.cnbc.com/2019/02/14/venezuela-crisis-bitcoin-trading-volumes-hit-an-all-time-high-.html.

[125] M.Girault and J. Misarsky. Selective forgery of RSA signatures using redundancy. In *Advances in Cryptology—EUROCRYPT'97: International Conference on the Theory and Application of Cryptographic Techniques*, 1997.

[126] G. Miller. Riemann's Hypothesis and Tests for Primality. In *Journal of Computer and System Sciences, 13(3)*, 1976.

[127] Usama Mir. Bitcoin and its energy usage: Existing approaches, important opinions, current trends, and future challenges. *KSII Transactions on Internet and Information Systems*, 14(8):3243–3256, Aug. 2020.

[128] A. Montag. Warren Buffett explains one thing people still don't understand about bitcoin, 2018. https://www.cnbc.com/2018/05/01/warren-buffett-bitcoin-isnt-an-investment.html.

[129] A. Montag. 'Wolf of Wall Street' Jordan Belfort on bitcoin: 'Get out if you don't want to lose all of your money', 2018. https://www.cnbc.com/2018/06/29/wolf-of-wall-street-jordan-belfort-get-out-of-bitcoin.html.

[130] Peter L Montgomery. Modular multiplication without trial division. *Mathematics of Computation*, 44(170):519–521, 1985.

[131] T. Moore, N. Christin, and J. Szurdi. Revisiting the Risks of Bitcoin Currency Exchange Closure. *ACM Transactions on Internet Technology*, 18(4), 2017.

[132] K. Moriarty, B. Kaliski, J. Jonsson, and A. Rusch. PKCS 1: RSA Cryptography Specifications Version 2.2, 2016.

[133] P. Nagel. Psychological Effects during Cryptocurrency Trading, 2018. http://www.scriptiesonline.uba.uva.nl/document/659099.

[134] S. Nakamoto. Bitcoin: A Peer-to-Peer Electronic Cash System, 2008. https://bitcoin.org/bitcoin.pdf.

[135] Shyam Narayanan. Improving the Speed and Accuracy of the Miller-Rabin Primality Test, 2014.

[136] R. Nieva. Ashes to ashes, peer to peer: An oral history of Napster, 2013. http://fortune.com/2013/09/05/ashes-to-ashes-peer-to-peer-an-oral-history-of-napster.

[137] J. O'Donovan, H. Wagner, and S. Zeume. The value of offshore secrets: Evidence from the Panama Papers. *The Review of Financial Studies*, 2019.

[138] H. Partz. Danish Man Faces Over 4 Years in Prison for Laundering 450K With Bitcoin, 2019. https://tinyurl.com/yymkx4b5.

[139] N. Pathak. What is IP blocklisting?, 2023. https://www.geeksforgeeks.org/what-is-ip-blocklisting.

[140] P. Pearce, B. Jones, F. Li, R. Ensafi, N. Feamster, N. Weaver, and V. Paxson. Global Measurement of DNS Manipulation. In *26th USENIX Security Symposium*, pages 427–443, Vancouver, Canada, 2017.

[141] K. Pietrzak. Simple verifiable delay functions. In *10th Innovations in Theoretical Computer Science Conference, ITCS 201*, 2019.

[142] J. Postel. Internet Protocol, 1981. https://datatracker.ietf.org/doc/html/rfc791.

[143] J. Postel. Transmission Control Protocol, 1981. https://www.ietf.org/rfc/rfc793.txt.

[144] Jean-Jacques Quisquater and Jean-Marc Couvreur. Fast decipherment algorithm for RSA public-key cryptosystem. *Electronics Letters*, 25(13):838–840, 1989.

[145] M. Rabin. Digitalized signatures and public-key functions as intractable as factorization. In *MIT/LCS/TR-212, MIT Laboratory for Computer Science*, 1979.

[146] G. Raimondo and L. Locascio. Module-Lattice-Based Digital Signature Standard (FIPS 204), 2023.

[147] G. Raimondo and L. Locascio. Module-Lattice-based Key-Encapsulation Mechanism Standard (ML-KEM) (FIPS PUB 203), 2023.

[148] G. Raimondo and L. Locascio. Stateless Hash-Based Digital Signature Standard (FIPS 205), 2023.

[149] R. Rivest, A. Shamir, and L. Adleman. A method for obtaining digital signatures and public-key cryptosystems. *Communications of the ACM*, 21(2):120–126, 1978.

[150] R. Rivest, A. Shamir, and L. Adleman. A method for obtaining digital signatures and public-key cryptosystems. *Communications of the ACM*, 1983.

[151] R. Rivest, A. Shamir, and D. Wagner. Time-lock puzzles and timed-release crypto. In *MIT/LCS/TR-684, MIT Laboratory for Computer Science*, 1996.

[152] K. Sako. An auction protocol which hides bids of losers. In *International Workshop on Public Key Cryptography*. Springer, 2000.

[153] W. Scheuerman. Whistleblowing as civil disobedience: The case of Edward Snowden. *Philosophy & Social Criticism*, 2014.

[154] A. Scott. 11 Countries Where Bitcoin Is Still Illegal, 2018. https://bitcoinist.com/11-countries-bitcoin-still-illegal.

[155] T.J. Seppala. The Pirate Bay shutdown: the whole story (so far), 2014. https://www.engadget.com/2014/12/16/pirate-bay-shutdown-explainer.

[156] Y. Seurin. On the lossiness of the Rabin trapdoor function. In *International Workshop on Public Key Cryptography '14*, pages 380–398, 2014.

[157] Adi Shamir. Identity-based cryptosystems and signature schemes. *Advances in Cryptology—CRYPTO'84*, 196:47–53, 1985.

[158] A. Smith and Y. Zhang. On the regularity of lossy RSA: Improved bounds and applications to padding-based encryption. In *Theory of Cryptography Conference '15*, pages 609–628, 2015.

[159] B. Smith. How To Track Illegal Funding Campaigns Via Cryptocurrency, 2019. https://www.bellingcat.com/resources/how-tos/2019/03/26/how-to-track-illegal-funding-campaigns-via-cryptocurrency.

[160] A. Tan and Y. Nakamura. Cryptocurrency Markets Are Juicy Targets for Hackers: Timeline, 2018. https://www.bloomberg.com/news/articles/2018-06-20/cryptocurrency-markets-are-juicy-targets-for-hackers-timeline.

[161] TCG. Trusted Computing Group Glossary, 2017.

[162] M.C. Tschantz, S. Afroz, and V. Paxon. SoK: Towards Grounding Censorship Circumvention in Empiricism. In *IEEE Symposium on Security and Privacy*, pages 914–933, San Jose, California, 2016.

[163] N. van Saberhagen. CryptoNote 2.0, 2013. https://github.com/monero-project/research-lab/blob/master/whitepaper/whitepaper.pdf.

[164] J. Verble. The NSA and Edward Snowden: surveillance in the 21st century. *Acm Sigcas Computers and Society*, 2014.

[165] W. Vickrey. Counterspeculation, auctions, and competitive sealed tenders. *The Journal of finance*, 1961.

[166] B. Wesolowski. Efficient Verifiable Delay Functions. In *Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security*, pages 251–266. ACM, 2019.

[167] P. Westfall. Kurtosis as Peakedness. In *The American Statistician*, 2014.

[168] A. Zainuddin. Coins, Tokens and Altcoins: What's the Difference?, 2017. https://masterthecrypto.com/differences-between-cryptocurrency-coins-and-tokens.

[169] Y. Zheng. Digital signcryption or how to achieve cost(signature and encryption) cost(signature) + cost(encryption). In *International Conference of Theory Applied Cryptography Technology (CRYPTO) 97*, pages 165 – 179. Springer, 1997.

[170] Y. Zheng. A new efficient signcryption scheme in the standard model. In *Security and Communication Networks vol. 8, no 5*, pages 703 – 878, 2015.

[171] Y. Zheng and H. Imai. How to construct efficient signcryption schemes on elliptic curves. In *Proc. of IFIP SEC98 vol. 68, no. 5*, pages 227 – 233, 1998.

[172] P. Zimmerman. Why I Wrote PGP, Essays on PGP. Phil Zimmermann and Associates LLC, 2013.