UNIVERSITÀ
DEGLI STUDI
DI PADOVA

Head Office: Università degli Studi di Padova
Administrative Office: Department of General Psychology

Doctorate Degree: Brain, Mind and Computer Science (BMCS)
Curriculum: Computer Science and Innovation for Societal Challenges
Cycle: XXXII

Security and Privacy Implications of Cryptocurrencies

Thesis written with the financial contribution of Fondazione Cassa di Risparmio di Padova e Rovigo.

Coordinator: Prof. Giuseppe Sartori
Supervisor: Prof. Mauro Conti
Co-Supervisor: Prof. Anna Spagnolli

Ph.D. Candidate: Ankit Gangwal

University of Padova

Human Inspired Technology Research Centre

Doctorate Degree: Brain, Mind and Computer Science
Curriculum: Computer Science and Innovation for Societal Challenges

---

Security and Privacy Implications of Cryptocurrencies

Candidate
Ankit Gangwal

Supervisor
Prof. Mauro Conti

Co-Supervisor
Prof. Anna Spagnolli

University of Padova, Italy

October 29, 2019

Acknowledgements

I am blessed to have had the support of several people in the journey of my Ph.D. Words are not enough to express my feelings for them. However, I would like to (putting mildly) thank five groups of people, without whom this thesis would not have been possible: my thesis advisor and committee members, my funding agencies, my family, my international mentors and collaborators, and my colleagues and friends.

I express my sincere gratitude to my Ph.D. advisor Prof. Mauro Conti for his constant motivation, valuable support, endless patience, and inexhaustible enthusiasm in guiding me in my research career. It is an honor working with such a talented, and yet, down to earth personality. I hope to keep learning from him in my future scientific career and personal life.

A sincere thank to my Ph.D. co-advisor Prof. Anna Spagnolli. Despite being from a different research field, she helped me in grooming my interdisciplinary knowledge. I would like to give a special thank to the members of my thesis committee and review board for their valuable comments and suggestions. I would also like to thank all the talented faculty members and friendly staff members associated with the BMCS Ph.D. school.

A huge thank to Fondazione Cassa di Risparmio di Padova e Rovigo (CaRiPaRo) for fully funding my Ph.D. program and allowing me to experience several unforgettable "bella-vita" moments.

Many thanks to my father Arvind Gangwal, my mother Usha Gangwal, and my brother Prateek Gangwal. They patiently supported me throughout my life and have always encouraged me to follow my ambitions. I could have never achieved this milestone in my life without their love and support. A special thank to my friend and fiancée Megha Gupta for her love, patience, and continuous encouragement to improve myself as a man.

I would genuinely like to thank Prof. Giuseppe Ateniese for being an extraordinary and inspiring mentor to me, especially during my visiting period at Stevens Institute of Technology, USA. I would also like to thank Prof. Massimo Bernaschi, Prof. Fabio Aiolli, Prof. Gabriele Tolomei, Prof. Sushmita Ruj, and Prof. Manoj Singh Gaur for accompanying me during the critical time of my Ph.D. program.

A big thanks to the PostDocs, Ph.D., M.Sc., and B.Sc. students that collaborated and exchanged knowledge with me, in particular, Mirko Polato, Eleonora Losiouk, Chhagan Lal, Sarada Prasad Gochhayat, Dario Pasquini, Md. Hassan Khan, Michele Todero, Md. Hajian Berenjestanaki, Gianluca Lain, and Samuele Giuliano Piazzetta.

I would like to thank my friends, colleagues, and officemates at the University of Padova (Riccardo Spolaor, Stefano Cecconello, Matteo Cardaioli, Giulio Rigoni, Daniele Lain, Luca Pasa, Moreno Ambrosin, and Roman Pukhtaievych) for their professional and personal help as well as for the amicable time we shared throughout these three years. Moreover, a sincere thank to the other members of the SPRITZ Research Group (Giuseppe Bernieri, Riccardo Lazzeretti, QianQian Li, and Piero Romare).

Last, but not least, I want to thank my loving friends (Nicola Pi Greco, Mario Sangiorgi, Valerio Mazzilli, Gabriele Albani, Mattia Lamon, Arianna Marini, Ambra Fastelli, Beatrice Moret, Marco Pignatelli, Davide Rigoni, Rossella Goffredo, and Zdravka Kirilova) for taking care of me and making me feel comfortable in Italy (a home away from home).

Ankit Gangwal

Padova, October 29, 2019

Abstract

Cryptocurrencies are cryptography-based digital currencies. In contrast to the traditional fiat currencies that are issued by centralized banking systems, cryptocurrencies are decentralized and maintained through distributed consensus mechanisms. The first truly functional cryptocurrency, i.e., Bitcoin, was introduced in November 2008 by Satoshi Nakamoto. Within a few years of its quiet launch, Bitcoin flourished to make a billion-dollar economy. After the massive success of Bitcoin, several other cryptocurrencies have been introduced to the market. As of September 2019, there are over 2500 active cryptocurrencies with more than $250 billion total market capitalization and nearly $50 billion daily volume.

Different cryptocurrencies work differently and aim to achieve different goals, e.g., some cryptocurrencies focus on limiting transaction throughput while others concentrate on performance. However, each cryptocurrency ensures a certain level of user anonymity. At the lowest level, users remain pseudo-anonymous, i.e., the real identities of payer and payee remain obscure. Consequently, cybercriminals have exploited the anonymity offered by cryptocurrencies in various crimes, including money laundering and terror financing. Moreover, cryptocurrencies bring several other severe concerns.

This thesis investigates the security and privacy implications of cryptocurrencies. This thesis is composed of three logical parts that focus on recently thriving, prominent, and severe concerns related to: (i) *Bitcoin*; (ii) *Algorand*; and (iii) *Cryptominers*.

In the first part of this thesis, we investigate two issues related to Bitcoin that hold significant importance in this era of cryptocurrencies. In particular, we focus on alarmingly increasing ransomware campaigns and the privacy concerns related to smartphone-based Bitcoin wallet apps. For the former, we present our comprehensive and longitudinal study on the recent ransomware attacks and report the economic impact of such ransomware from the Bitcoin payment perspective. For the latter, we present our work on identifying sensitive user activities on Bitcoin wallet apps that are commonly used for sending, receiving, and trading Bitcoin.

The second part of this thesis focuses on Algorand. Algorand is a truly democratic blockchain consensus protocol that has the potential to shape the future of blockchain technology. To the best of our knowledge, it is the first formal study on Algorand. In our security analysis, we propose a practically feasible attack on Algorand and its possible countermeasures.

In the third part of this thesis, we explore covert cryptomining. The demand for cryptomining has increased drastically with the increasing popularity of cryptocurrencies. In parallel to legitimate cryptomining demands, covert cryptomining has emerged as a utility for malicious actors to gain financial incentives. Cryptocurrencies, such as Monero, have further aggravated the situation by enabling even naive users to mine via a browser application. Considering the severity of the issue, we propose two efficient solutions to detect covert cryptomining under different real-world scenarios.

Contents

Chapter 1

Introduction

Cryptocurrencies are virtually existing digital assets that use cryptography to secure financial transactions, control the flow of new monetary units, and verify the transfer of assets. In contrast to the conventional centralized banking systems, cryptocurrencies are decentralized systems. These decentralized systems employ a distributed ledger technology, called *blockchain*, which serves as an append-only public database of the financial transactions. New transactions are added to the ledger only after verification via a process called *cryptomining*. The network nodes participating in the verification process are called *cryptominers*.

The fundamental concept of an untraceable payment system dates back to the '80s. D. Chaum [52] presented the idea of creating untraceable payments in electronic payment systems in 1983. In 1993, researchers from Carnegie Mellon University [203] and University of Southern California [148] discussed the need of a cryptography-based digital currency. Nevertheless, the idea of the first widely accepted and truly functional cryptocurrency, i.e., Bitcoin [156], was introduced in November 2008 by a person or a group of people under a pseudonym Satoshi Nakamoto. Bitcoin cleverly weaves already existing knowledge derived from decades of research. Within a few years of its quiet launch, Bitcoin flourished to make a billion-dollar economy.

After the success of Bitcoin, several other cryptocurrencies have been introduced to the market. These new alternative cryptocurrencies are called *Altcoins*. As of September 2019, there are over 2500 active cryptocurrencies with over $250 billion total market capitalization and nearly $50 billion daily volume [2]. JPMorgan Chase - a major financial institution - has recently announced its digital coin for payments [122]. Several online shopping websites, supermarkets, etc. nowadays accept cryptocurrency as a valid mode of payment. Moreover, an increasing number of users are investing and trading cryptocurrencies as regular commodities. To summarize, cryptocurrencies are dramatically affecting and infusing in our day-to-day lives.

1.1 Research Motivation and Contribution

Different cryptocurrencies aim to achieve different goals. For instance, some cryptocurrencies focus on controlling the throughput of transactions while others concentrate on performance. Nevertheless, users' anonymity remains the key feature across all the cryptocurrencies. The level of anonymity offered by a cryptocurrency depends on its underlying design principles. At the lowest level, users remain pseudo-anonymous, i.e., the real identities of payer and payee remain obscure. Intuitively, cybercriminals have exploited cryptocurrencies for several illegal activities [190], e.g., terror financing [88], money laundering [202], cyber-attacks [172] (in particular, ransomware [59]), and to sell prohibited items over the darknet (e.g., *Silk Road* [204] and *Black Market Reloaded* [102]). The pseudo-anonymity offered by cryptocurrencies brings several severe concerns. In this thesis, we investigate the security and privacy implications of cryptocurrencies. We present our research work in the following three logical parts:

1. *Bitcoin* focuses on understanding Bitcoin-based ransomware campaigns and privacy concerns related to smartphone-based Bitcoin wallet apps.
2. *Algorand* investigates a practically feasible attack on Algorand, which is an innovative and truly democratic blockchain consensus protocol.
3. *Cryptominers* presents two efficient solutions to detect covert cryptomining under different real-world scenarios.

In what follows, we briefly introduce the parts mentioned above and highlight our contributions. In this thesis, some passages have been quoted verbatim, and some figures have been reused from the works [34, 58–60, 104], all co-authored by the author of this thesis.

1.1.1 Bitcoin

Satoshi Nakamoto introduced Bitcoin in 2008, which is by far the most successful peer-to-peer, decentralized, pseudo-anonymous, and cryptography-based electronic currency [156]. Its source code was released as open-source software in 2009 [157]. Bitcoin enables users to transact securely and pseudo-anonymously by using an arbitrary number of aliases (Bitcoin addresses). Bitcoin intelligently combines already existing works derived from decades of research. Bitcoin made a billion-dollar economy within a few years of its launch. Such a success of this pseudo-anonymous currency attracted cybercriminals to exploit the Bitcoin system in several different ways. Moreover, it also comes with privacy concerns for the users. In Part I of this thesis, we focus on two significantly important issues related to Bitcoin. In particular, we investigate alarmingly increasing ransomware attacks and the privacy concerns for the smartphone-based Bitcoin wallet app users.

Bitcoin Ransomware Campaigns

Ransomware is a class of malware that restricts access to the system it infects until the victim pays the demanded ransom. According to a report by Symantec Inc. [195], ransomware continued to be the most dangerous cybercrime threat to individual users and enterprises. Moreover, ransomware attacks are becoming alarmingly frequent. Most recently, several schools and universities were targeted by ransomware attacks across the United States of America, which delayed the start of the year [161]. The evolving class of ransomware has been exploiting privacy-preserving online services (e.g., Tor hidden network [87]) to remain anonymous. Furthermore, the pseudo-anonymous nature of decentralized cryptocurrencies (e.g., Bitcoin) makes it difficult to trace a payee. Hence, the cybercriminals have been misusing cryptocurrencies to extort ransoms anonymously. Being the most commonly known cryptocurrency, Bitcoin is exploited the most by ransomware.

Contributions: In Chapter 2, we present our comprehensive and longitudinal study on recent ransomware attacks and report the economic impact of such campaigns from the Bitcoin payment perspective. We also present a lightweight framework to identify, collect, and analyze Bitcoin addresses managed by the same user or group of users (cybercriminals, in this case), which includes a novel approach for classifying a payment as ransom. To verify the correctness of our framework, we compared our findings on CryptoLocker ransomware with the results presented in the literature. Our results align with the results found in the previous works except for the final valuation in USD. The reason for this discrepancy is that we used the average Bitcoin price on the day of each ransom payment whereas the authors of the previous studies used the Bitcoin price on the day of their evaluation. Furthermore, for each investigated ransomware, we provide a holistic view of its genesis, development, the process of infection and execution, and characteristic of ransom demands. Finally, we make our collected datasets and knowledge-base publicly available.

Privacy Issues in Bitcoin Wallet Apps

Smartphones have become an integral part of our daily lives. Global smartphone sales have increased by nearly 1200% [160] in the last decade. Generally, smartphones come with standard tools and utilities out of the box. Nevertheless, a user can download and install additional applications, called apps, to get supplementary features in his/her smartphone. On another side, Bitcoin has emerged as one of the most promising means for payments, remittance, and trading. Supplemented by the convenience offered by smartphones, an increasing number of users are adopting Bitcoin wallet apps for different purposes.

Contributions: In Chapter 3, we focus on identifying user activities on smartphone-based Bitcoin wallet apps that are commonly used for sending, receiving, and trading Bitcoin. To accomplish our goal, we perform network traffic analysis using machine learning techniques. Since we focus on apps of the same type/functionality, it makes our classification problem even more difficult compared to classifying apps tailored for discrete purposes. Moreover, our goal is to identify user activities even in the presence of encryption. In our experiments, we considered the worldwide most downloaded Bitcoin wallet apps on both Google Play Store and Apple's App Store. We used only physical hardware and omitted any emulator to build our experiment scenario as close to the real environment as possible. We process the traffic traces in several phases before extracting the features that are utilized to train our supervised learning algorithms. We deal with the classification problem in multiple stages in a hierarchical fashion. In our experiments, our system attained nearly 95% accuracy in user activity identification.

1.1.2 Algorand

A variety of solutions, e.g., Proof-of-Work (PoW), Proof-of-Stake (PoS), Proof-of-Burn (PoB), and Proof-of-Elapsed-Time (PoET), have been proposed to make consensus mechanism used by the blockchain technology more democratic, efficient, and scalable. However, these solutions have a number of limitations, e.g., PoW approach requires a huge amount of computational power, scales poorly, and wastes a lot of electrical energy. Recently, an innovative and truly democratic protocol Algorand [54] has been proposed to overcome these limitations. Algorand aims to solve the "blockchain trilemma" of decentralization, scalability, and security. In Part II of this thesis, we analyze this truly democratic blockchain consensus protocol from a security point of view.

Security Flaw in Truly Democratic Consensus Protocol

Algorand uses a process, called *cryptographic sortition*, to securely and unpredictably elect a set of voters from its network periodically. These voters are responsible for reaching consensus through a Byzantine Agreement (BA) protocol for one block at a time, which guarantees an overwhelming probability of the blockchain's linearity and a block generation time of nearly a minute. It is comparable to the works [40, 131] and has the potential to shape the future of blockchain technology. Given the promising properties of Algorand, its security aspects are crucial.

Contributions: In Chapter 4, we present a security analysis of Algorand. To the best of our knowledge, it is the first security analysis as well as the

first formal study on Algorand. We designed an attack scenario in which a group of malicious users tries to break the protocol, or at least limits it to a reduced partition of network users, by exploiting a security flaw in the messages validation process of the BA protocol. Since the source code or an official simulator for Algorand was not available at the time of our study, we created a simulator to implement the protocol and assess the feasibility of our attack scenario. Our attack requires the attacker to merely have the trivial capability of establishing multiple connections with targeted nodes, and it costs practically nothing to the attacker. Our results show that it is possible to slow down the message validation process on honest nodes, which eventually forces them to select default values on the consensus, leaving the targeted nodes behind in the chain as compared to the non-attacked nodes. Even though our results are subject to the real implementation of the protocol, the core concept of our attack remains valid.

1.1.3 Cryptominers

Cryptomining is a process of validating and adding new transactions in the blockchain digital ledger for a given cryptocurrency. It is an essential process to keep most of the cryptocurrencies running. With an increasing number of cryptocurrencies, the demand for cryptomining has increased notably. This demand continues to remain huge because cryptomining, as mentioned before, is an inevitable operation to keep these currencies running. Such a huge demand for mining has also attracted cybercriminals to earn financial gains via covert cryptomining [11, 28]. Covert cryptomining is defined as an unauthorized harnessing of computational resources to mine cryptocurrencies. In fact, unauthorized cryptomining attacks exceeded ransomware attacks in 2018 and affected five times more systems [33]. Such exploitation of the computational resources causes financial damage - primarily in the form of increased electricity bills - to the victims, who often discover the misuse when the damage has already been done. Additionally, prolonged mining on an incompatible device may also harm the hardware [12]. In Part III of this thesis, we investigate covert cryptomining and propose two practical solutions to detect it under different real-world scenarios.

Detecting Covert Miners via Magnetic Side-channel

We can broadly categorize covert cryptominers into two classes: (1) conscious-miners and (2) unconscious-miners. Conscious-miners (e.g., an unethical employee) exploit infrastructure allocated to them. Unconscious-miners (e.g., a visitors to a website that hosts cryptomining scripts) mine unknowingly for a third party. The task of cryptomining is to execute the core mining algorithm repeatedly, which means that a robust signature can be constructed for a particular algorithm. Interestingly, there are a limited number of min-

ing algorithms. Hence, we focus on the mining algorithms. Notably, such signature-based detection can be partially deceived by restricted cryptomining. But, it would directly affect the hashing rate and consequently the profits; making the task of cryptomining less appealing.

Contributions: In Chapter 5, we present a novel approach that leverages magnetic side-channel to detect both types of covert cryptominers. Our proposed approach merely requires the physical proximity of the examiner and a magnetic sensor, which is often available on smartphones, and it works even when the examiner does not have login-access or root-privileges on the suspect device. The fundamental idea of our approach is to profile the magnetic field emission of a processor for the set of available mining algorithms. We built a complete implementation of our system using advanced machine learning techniques. In our experiments, we included all the cryptocurrencies supported by the top-10 mining pools, which collectively comprise the largest share (84% during Q3 2018) of the market. By using the data recorded from the magnetometer of an ordinary smartphone, our classifier achieved an average precision of over 88% and an average F_1 score of 87%.

Detecting Covert Miners via Hardware Performance Counters

In continuation of our work on detecting covert cryptomining, we propose an approach that is designed for another critical real-world scenario. Our proposal in Chapter 5 uses magnetic side-channel to detect both types of covert cryptomining. In that case, we assume that the examiner does not have login-access (corresponds to conscious-mining) or root-privileges (corresponds to unconscious-mining) on the suspect device. Now, we specifically focus on unconscious-mining, and unlike the previous case, the examiner here owns the device and can elevate his/her access privileges on the device.

Contributions: In Chapter 6, we present an efficient and generic approach to detect covert cryptomining on personal computers. We utilize Hardware Performance Counters (HPC) to create signatures that grasp the execution pattern of mining algorithms on a processor. We evaluated our methodology on two different processors through an exhaustive set of experiments. In our experiments, we considered all the cryptocurrencies supported by the top-10 mining pools. Our results show that our approach can achieve a near-perfect classification performance with samples of length as low as five seconds. Due to its robust design, our solution can even adapt to zero-day cryptocurrencies. We believe that our solution is practical and can be deployed to tackle the uprising problem of covert cryptomining.

Finally, we draw conclusions and possible future works in Chapter 7.

1.2 Publications

The research works presented in this thesis and done during my Ph.D. program produced peer-reviewed journal, conference, and workshop publications. A complete list of the published and currently submitted papers is listed in chronological order as follows: Section 1.2.1 lists the journal papers and Section 1.2.2 lists the conference and workshop papers.

1.2.1 Journal Publications

[J1] M. Conti, A. Gangwal, and S. Ruj. "On the Economic Significance of Ransomware Campaigns: A Bitcoin Transactions Perspective." In *Elsevier Computers & Security*, 79: 162-189, 2018. **(JCR IF 2017: 2.650; IT-ANVUR Class: 2)**

[J2] A. Gangwal and M. Conti. "Cryptomining cannot Change its Spots: Detecting Covert Cryptomining using Magnetic Side-channel." In *IEEE Transactions on Information Forensics & Security*, in press, 2019. **(JCR IF 2018: 6.211; IT-ANVUR Class: 1)**

[J3] M. Conti, A. Gangwal, G. Lain, and S. G. Piazzetta. "Detecting Covert Cryptomining using Hardware Performance Counters." Under review at *IEEE Transactions on Dependable & Secure Computing*, 2019.

[J4] M. Conti, A. Gangwal, M. Hassan, C. Lal, and E. Losiouk. "The Road Ahead for Networking: A Survey on ICN-IP Coexistence Solutions." Under review at *IEEE Communications Surveys & Tutorials*, 2019.

1.2.2 Conference and Workshop Publications

[C1] A. Gangwal, M. Conti, and M. S. Gaur. "PANORAMA: Real-time Bird's Eye View of an OpenFlow Network." In *Proceedings of the 14th IEEE International Conference on Networking, Sensing and Control (ICNSC)*, pages 204-209, 2017.

[C2] M. Conti, A. Gangwal, and M. S. Gaur. "A Comprehensive and Effective Mechanism for DDoS Detection in SDN." In *Proceedings of the 13th IEEE International Conference on Wireless and Mobile Computing, Networking and Communications (WiMob)*, pages 1-8, 2017. **(Acceptance rate: 28%)**.

[C3] M. Conti and A. Gangwal. "Blocking Intrusions at Border using Software Defined-Internet Exchange Point (SD-IXP)." In *Proceedings of the 3rd IEEE Conference on Network Functions Virtualization and Software Defined Networking (NFV-SDN)*, pages 1-6, 2017.

[C4] M. Conti, A. Gangwal, S. P. Gochhayat, and G. Tolomei. "Spot the Difference: Your Bucket is Leaking - A Novel Methodology to Expose A/B Testing Effortlessly." In *Proceedings of the 4th IEEE Workshop on Security and Privacy in the Cloud (SPC) @ IEEE CNS*, pages 1-7, 2018.

[C5] F. Aiolli, M. Conti, A. Gangwal, and M. Polato. "Mind Your Wallet's Privacy: Identifying Bitcoin Wallet Apps and User's Actions through Network Traffic Analysis." In *Proceedings of the 34th ACM Symposium on Applied Computing (SAC)*, pages 1484-1491, 2019.

[C6] M. H. Berenjestanaki, M. Conti, and A. Gangwal. "On the Exploitation of Online SMS Receiving Services to Forge ID Verification." In *Proceedings of the 14th International Conference on Availability, Reliability and Security (ARES)*, pages 1-5, 2019. **(Acceptance rate: 20.75%)**.

[C7] M. Conti, A. Gangwal, and M. Todero. "Blockchain Trilemma Solver Algorand has Dilemma over Undecidable Messages." In *Proceedings of the 14th International Conference on Availability, Reliability and Security (ARES)*, pages 1-8, 2019. **(Acceptance rate: 20.75%)**.

[C8] D. Pasquini, A. Gangwal, G. Ateniese, M. Bernaschi, M. Conti. "Improving Password Guessing via Representation Learning." Under review at *IEEE Symposium on Security and Privacy (S&P)*, 2020.

Part I

Bitcoin

Chapter 2

Bitcoin Ransomware Campaigns

Satoshi Nakamoto in 2008 proposed a decentralized and cryptography-based electronic currency called Bitcoin [156]. Such financial systems eliminate the control of centralized authority and provide ubiquity as well as fairness via (quasi) real-time transactions. Such digital currencies also guarantee a certain degree of anonymity, which raises novel and unique concerns, e.g., an inevitable-growth in illegal activities.

On another side, ransomware is a class of malware that restricts access to the system it infects until the victim pays the demanded ransom. Readily available toolkits such as eda2[1] and Ransomware-as-a-Service (RaaS) enable even a novice user to create and launch ransomware. Furthermore, the ransomware affiliate program lures users to spread ransomware in exchange for profit share. According to the annual threat report-2017 published by Symantec Inc. [195], ransomware continued to be the most dangerous cybercrime threat to individual users and enterprises in 2016. Compared to the previous year, the number of detected ransomware infection increased by 36% during 2016. Moreover, average ransomware detection rate reached over 1,500 incidents per day at the year-end. In particular, the average ransom amount rose 266% from USD 294 in 2015 to USD 1,077.

The evolving class of ransomware has been exploiting privacy-preserving online services, e.g., the Tor hidden network [87] to remain anonymous. Moreover, the pseudo-anonymous nature of decentralized currencies such as Bitcoin makes it difficult to trace a payee. Hence, the cybercriminals have been misusing such payment systems to extort ransoms anonymously. In this chapter, we present our comprehensive and longitudinal study on recent ransomware and report the economic impact of such ransomware from the Bitcoin payment perspective.

[1] eda2 is an abandoned open-source ransomware kit that was distributed only for educational purposes.

Contributions: Our contributions to the state-of-the-art are as follows:

1. We present a lightweight framework to identify, collect, and analyze addresses that belong to the same user. We also propose a novel approach for classifying a payment as ransom.
2. Using our framework, we analyzed the economic impact (in terms of ransoms extorted in Bitcoin) of all the recent ransomware: (i) that used Bitcoin as at least one mode of ransom payment and (ii) for which at least one Bitcoin address is publicly known.
3. We discuss the inception, evolution (where applicable), and functionality (including distribution, infection, and encryption procedure) of every analyzed ransomware along with the magnitude and timeline of their ransom demands.
4. We also release our dataset[2] for future research endeavors. The dataset contains a detailed transaction history of all the addresses we identified for each ransomware. Hence, our results are fully reproducible.

To the best of our knowledge, our work is the first study that not only elaborates the characteristics and functionality of various Bitcoin ransomware, but it also gives more accurate insights on the economic impact of such ransomware. In particular, our work is different from the state-of-the-art on the following dimensions:

1. Unlike existing works [120,121,130,138,190], we consider both the day-to-day lowest and highest Bitcoin price as well as the variations due to the transaction fee to identify a payment as ransom.
2. To accurately assess the worth (in USD) of extorted ransoms, we use the average Bitcoin price on the day of each ransom payment.
3. Our framework focuses on the transactions belonging to the address(es) of interest rather than the entire blockchain, which saves bandwidth, storage, and computational resources while querying the blockchain.

Organization: The rest of the chapter is organized as follows: Section 2.1 elucidates the essential concepts related to ransomware infection and the Bitcoin currency system. Section 2.2 addresses the previous works on identification and assessment of cybercrimes in the Bitcoin ecosystem. In Section 2.3, we explain our framework for ransom identification. In Section 2.4, we present our findings and enlighten the economic impact the ransomware that fulfilled our selection criteria. In Section 2.5, we discuss the limitations of our proposed framework. Finally, Section 2.6 summarizes the chapter.

[2]https://tinyurl.com/y4u3c7zr

2.1 Background & Preliminaries

In this section, we describe the chronology of a typical ransomware infection and explain the fundamentals of the Bitcoin cryptocurrency system.

Ransomware: A typical ransomware infection includes the following events:

1. Infection: Similar to generic malware, ransomware are also distributed via various infection vectors. These vectors include, but not limited to, email spamming with malicious attachment (e.g., CryptoLocker) or link to the malicious payload (e.g., CryptoWall), exploit packs (e.g., Angler browser exploit in TeslaCrypt and Neutrino exploit kit in DMA Locker). Interestingly, recent ransomware incorporate self-propagation capabilities. For instance, NotPetya and WannaCry exploit vulnerabilities in the network protocols to infect local computers on the same network.

2. Encryption: After infiltration, ransomware silently encrypt files on the infected system. In particular, ransomware target those files that are valuable to the user, e.g., images, videos, documents. For the encryption process, ransomware use symmetric encryption algorithm, asymmetric encryption algorithm, or even combination of the both. The key for encryption is either generated locally or procured from a remote Command and Control (C&C). Generally, the backup files are also encrypted/deleted to prevent recovery. However, the files responsible for running the system are not affected, at least until the deadline for the ransom payment.

3. Extortion: After the encryption process, ransomware typically display a ransom note on the screen. Generally, the ransom note of recent ransomware includes a threat message, ransom amount specified in fiat currency such as US dollar (for instance, USD 300 in NotPetya) or cryptocurrency such as Bitcoin (for instance, 1 BTC in CryptoLocker), a countdown timer that shows the time left before the deadline, and a payment address. The payment address can be a Bitcoin address or a website's address that shows this Bitcoin address. The ransom note often includes instructions on how and where to buy Bitcoin.

4. Decryption: After confirmation of the ransom payment, the ransomware either automatically start the decryption process, or the victim is asked to download and run a decryption tool.

Bitcoin: D. Chaum in his work [52] introduced the idea of untraceable payments. Soon after, researchers from Carnegie Mellon University [203] and University of Southern California [148] scrutinized the necessity for a

cryptography-based digital currency. In November 2008, Satoshi Nakamoto articulated a peer-to-peer, decentralized, cryptography-based electronic currency system called Bitcoin [156]. The basic terminologies used in the Bitcoin system are:

- Address: A Bitcoin address is a string identifier of a possible destination for a Bitcoin payment. It is 26 to 35 alphanumeric characters long and begins with the number 1 (Pay-to-Pub KeyHash or P2PKH type) or 3 (Pay to Script Hash or P2SH type). Bitcoin addresses are hashed public keys generated from the Elliptic Curve Digital Signature Algorithm (ECDSA). Hence, each Bitcoin is associated with the owner's public key.

- Wallet: A wallet is a file that stores Bitcoin addresses along with the corresponding private keys. It also maintains the Unspent Transaction Output (UTXO) corresponding to each address.

- Blockchain: The blockchain is a shared, public ledger on which the entire Bitcoin network relies. All confirmed transactions are included in the blockchain without any exception. This way, new transactions can be verified to be spending Bitcoin that are indeed owned by the spender. The integrity and the chronological order of the blockchain are enforced with cryptography.

- Block: An individual unit of the blockchain is called a block. Each block includes the hash of the previous block to guarantee the integrity of the network, the nonce that assisted its mining, and a list of the transactions.

- Transaction: A transaction refers to a transfer of Bitcoin between Bitcoin addresses. To transfer Bitcoin, a payer creates a transaction message. In this message, the payer specifies the payee's Bitcoin address as well as an amount of Bitcoin to transfer. As shown in Figure 2.1, the payer authenticates the transaction by digitally signing it with the private key of the corresponding address. Finally, Bitcoin network broadcasts and confirms (typically, in the following 10 minutes) the transaction through a process called mining. A confirmed transaction is irreversible.

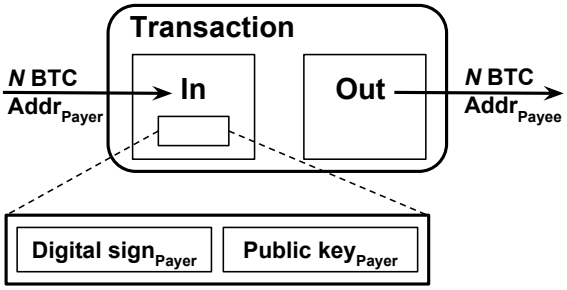


Figure 2.1: An example of a simple Bitcoin transaction

A user can also purchase Bitcoin in exchange for other regulated currencies. The unit of the Bitcoin currency is *Bitcoin*, abbreviated as BTC. Like any other traded commodity, the price[3] of Bitcoin varies. Figure 2.2 depicts the BTC-USD exchange rate since July 18, 2010, the day when one of the world's first Bitcoin currency exchange market Mt. Gox was established.

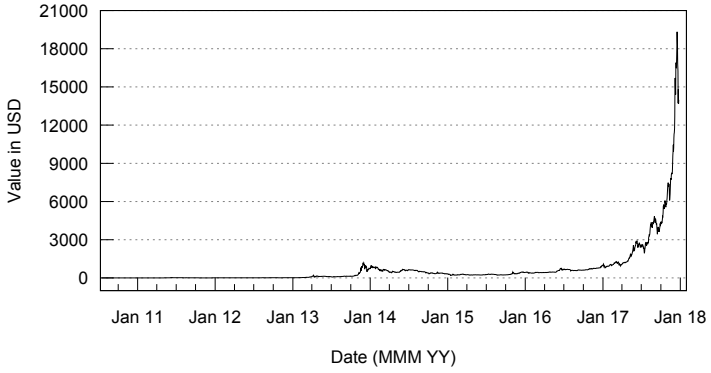


Figure 2.2: BTC-USD exchange rate trend

2.2 Related Works

Law enforcement authorities as well as the research community have made several attempts to identify and measure cybercrimes in the Bitcoin ecosystem. The authors in [44,86,147,175] proposed tools to analyze transactions in the Bitcoin blockchain visually. Christin in [56] proposed a thorough analysis of the Silk Road anonymous marketplace and discussed the socio-economic implications of the findings. Ron and Shamir used the public blockchain data to estimate the wealth of the Silk Road marketplace's owner, known as Dread Pirate Robert [178]. Soska and Christin studied anonymous online

[3]We use the term "price" to refer BTC-USD exchange rate.

marketplaces including Silk Road, Sheep Marketplace, etc. and examined how virtual marketplaces have evolved [189]. Meiklejohn et al. [149] proposed an approach to comprehend overall transaction patterns of the Bitcoin payments used for criminal or fraudulent purposes.

However, the literature on measuring the economic impact of ransomware that accepted ransoms via Bitcoin (hereinafter referred to as "Bitcoin ransomware") is rather limited. Huang et al. [119] discuss the ethical and technical issues of monitoring ransomware activities as well as the dynamics of ransom payments. Liao et al. [138] analyzed the timestamps of ransom payments to CryptoLocker. The work [130] provides a holistic view of the general ransomware that appeared between 2006 and 2014. Additionally, the authors also estimated the financial intensives gained by CyptoLocker ransomware. Spagnuolo et al. proposed a framework called BitIodine [190]. The authors used BitIodine to investigate Bitcoin addresses associated with CryptoLocker ransomware and Dread Pirate Roberts. The works [120, 121] present a systematic analysis of CryptoLocker ransomware.

It is noteworthy that previous works [120, 121, 130, 138, 190] only considered either the daily average or highest Bitcoin price to classify ransom payments and do not take into account the variations that might occur due to the transaction fee. Furthermore, their estimation of the total worth of extorted ransoms is based on the Bitcoin price on the day of their evaluation, which exaggerates the results due to fluctuations (mostly, increase; see Figure 2.2) in the price of Bitcoin. Additionally, the systems proposed in the previous works [44, 86, 147, 175, 190] demand high bandwidth, storage, and computational resources as they query the entire blockchain.

2.3 Ransom Identification Framework

To investigate the ransoms extorted by a ransomware, we first identify the Bitcoin addresses linked to the ransomware. Then, we obtain the transaction history of these addresses. Finally, we distinguish the transactions associated with the ransom payments. To this end, we propose our framework, which consists of three stages/parts/modules: (i) identifying the Bitcoin addresses belonging to the ransomware (discussed in Section 2.3.1); (ii) data (transaction history) collection and database generation from the blockchain (presented in Section 2.3.2); and (iii) our considerations for classifying a payment as ransom (elaborated in Section 2.3.3).

2.3.1 Module 1: Identification of Ransomware Addresses

Bitcoin offers privacy only through pseudonymity, and an increasing number of works [43,149,175,177,178,189] suggest that information available in public blockchain ledger can lead to de-anonymize (to a certain extent) Bitcoin transactions.

To collect the addresses associated with a ransomware, we began by extensively searching various online resources: ransomware knowledge base (e.g., ESET, Kaspersky Lab, Malwarebytes, Symantec); ransomware removal guides (e.g., MalwareTips.com, 2-spyware.com, BleepingComputer.com, "How To" videos on YouTube); reports from Counter Threat Units (CTU), Incident Responses (IR), and Security Operations Centers (SOC) (e.g., Dell SecureWorks, PhishMe.com); online fora (e.g., Reddit) where victims and researchers post Bitcoin addresses associated with the concerned ransomware; and screenshots of ransomware available in different image search engines (e.g., Google, Yahoo). Considering the fact that not every address related to a ransomware is posted on the Internet, we used two clustering heuristics to identify the set of addresses controlled by the same user (cybercriminals, in our case). Our heuristics are based on the fundamental principles of the Bitcoin transaction protocol [156] and are as follows:

Multi-input transactions

A multi-input transaction usually[4] takes place when a user U attempts to make a payment, and the payment amount P cannot be sufficiently funded by any of the individual Bitcoin balance available in U's wallet. In such a scenario, the Bitcoin protocol allows grouping of a set of Bitcoin balances from U's wallet to settle P and make payment through a multi-input transaction. Hence, we can conclude that if a set of input addresses S_{input} is used to disburse P, then S_{input} is managed by the same user.

Shadow/change address

In the Bitcoin protocol, the whole input amount must be spent in the same transaction. To deliver the "change" back to the user U, a shadow address A_{shadow} is automatically generated and used to collect the unspent amount of the transaction. If there are two addresses in the set of output addresses S_{out}, and one address has never been seen before in the whole blockchain while the other address has appeared before, then we can safely presume that the newly generated address is a shadow address [149].

Algorithm 1 explains our approach to identify the addresses managed by the same user, hereinafter referred to as "Cluster". Here, $S_{initial}$ represents the set of addresses collected from the online resources, S_{input} is a set of input addresses in a transaction, and A_{shadow} represents a shadow address generated (if any) in a transaction.

[4]Nowadays, coin mixing services allow users to join their transactions to enhance anonymity and unlinkability. However, such services have many security and privacy concerns [61]. Hence, for simplicity, we assume that the user commonly does not make use of Bitcoin mixers.

Algorithm 1 Identifying addresses managed by the same user.

Input: $S_{initial}$

$Cluster := S_{initial}$
$Cluster' := \{\}$ ▷{ } is an empty set
while $Cluster \neq Cluster'$ **do**
  $Cluster' := Cluster$
  $M := \{\}$ ▷M stores S_{input}
  $C := \{\}$ ▷C stores A_{shadow}
  for i in $Cluster$ **do**
    Get all transactions Tx where i is an input address
    for t in Tx **do**
      $M \cup (S_{input}\ in\ t)$ ▷$\cup$ is set union
      $C \cup (A_{shadow}\ in\ t)$
    end for
  end for
  $Cluster := Cluster \cup M \cup C$
end while
return $Cluster$

Essentially, for a given list of addresses, our algorithm recursively finds all the addresses satisfying our heuristics.

2.3.2 Module 2: Data Collection and Database Generation

As explained in Section 2.2, Bitcoin blockchain data is publicly available. At the time of writing (December 2017), block height of the blockchain was over 500,000 blocks, which means that downloading/querying the entire blockchain is very expensive in terms of bandwidth, storage, and computations. To address these issues, we built a lightweight system that uses *Blockchain Data API*[5] to crawl and parse transactions associated only with the address(es) of interest.

For each transaction associated with an address of interest (*Address*), our system collects the hash of the transaction (*HASH*), remitted Bitcoin (*BTC_to_Addr*), GMT date (*GMT_Date*), and GMT time (*GMT_Time*), input addresses (*Trx_In_Addrs*), and output addresses (*Trx_Out_Addrs*). Listing 2.1 shows the SQL statement used to create our database.

[5]https://tinyurl.com/y6ssesr2

```
CREATE TABLE tx (
HASH CHAR(64) NOT NULL PRIMARY KEY,
BTC_to_Addr INT NOT NULL,
Trx_In_Addrs TEXT,
Trx_Out_Addrs TEXT,
GMT_Date DATE,
GMT_Time Time,
Address CHAR(35) NOT NULL,
Address_as_Input INT NOT NULL
);
```

Listing 2.1: SQL statement for creating our database

The field *HASH* serves as the *Primary Key*, which implicitly discards any duplicate transactions reported for multiple participating/constituting addresses. *Address_as_Input* denotes if the *Address* was used as an input in the transaction. Our system also uses *BitcoinAverage API*[6] to collect day-to-day highest, average, and lowest price of Bitcoin.

2.3.3 Module 3: Classifying a Payment as Ransom

A Bitcoin transaction involves two varying factors: (i) Bitcoin price, and (ii) transaction fee. The price of Bitcoin changes frequently. Therefore, considering only the daily average, highest, or lowest price of Bitcoin is not suitable, especially when the variation in the price is high. Furthermore, the transaction fee is paid on the top of the transaction amount. A victim may assume that the ransom amount to be paid includes (or excludes) the transaction fee, which leads to discrepancies in the payment-amount transferred to an address. Moreover, the transaction fee depends on the size of the transaction, i.e., a transaction that involves a larger number of addresses would incur more fee than a transaction with fewer addresses involved. Hence, to classify a payment as ransom, our framework considers both the day-to-day lowest and highest price of Bitcoin as well as the variation that might occur due to the transaction fee.

In general, the cybercriminals specify the ransom either in Bitcoin (e.g., 1 BTC) or USD equivalent BTC (e.g., Bitcoin equivalent to USD 300). Our framework classifies a payment ρ to an address α in a transaction τ as ransom if it satisfies at least one condition in Eq. 2.1a or Eq. 2.1b.

[6]https://tinyurl.com/ybzkbg7k

$$demand\ in = \begin{cases} BTC = \begin{cases} r_b = d_b, \\ r_b = d_b - f, \end{cases} & \text{(2.1a)} \\ USD = \begin{cases} v_l \leq d_u \leq v_h, \\ v_l \leq d_u - f \leq v_h, \end{cases} & \text{(2.1b)} \end{cases}$$

where:

- f denotes the transaction fee, computed as the difference between the total amount being spent and the total amount being received in τ.
- d_b denotes the ransom asked in BTC.
- d_u denotes the ransom asked in USD.
- r_b denotes the BTC received by α in ρ.
- v_l denotes the value of r_b computed using the lowest BTC price of the payment day.
- v_h denotes the value of r_b computed using the highest BTC price of the payment day.

It is also important to mention that to evaluate the total ransom (in USD) received by a ransomware cluster, it would be unfair to use the Bitcoin price on the day of our evaluation as it would misrepresent the amount due to the variations in the price. Hence, unlike previous works, we used the average Bitcoin price on the day of each ransom payment.

2.4 Economic Impact of Ransomware

We found twenty ransomware that fulfilled our selection criteria, i.e., those ransomware: (i) that used Bitcoin as at least one mode of ransom payment, and (ii) for which at least one Bitcoin address is publicly known. Figure 2.3 depicts the reported debut period of these ransomware as well as the occurrence of their renamed/rebranded versions. We performed the numerical assessment of the ransomware on December 7, 2017. Hence, all the data reported in this chapter include the transactions until December 7, 2017. In this chapter, we discuss only those ransomware for which the observed payments align with their period of activity and ransom demands. The details of other ransomware have been omitted in this chapter to maintain the focus of this thesis. The details of such ransomware can be found in our work [59]. Table 2.1 presents a summary of overall payments received by the addresses of ransomware presented in this chapter. It also lists the payments classified as ransom by our framework. Furthermore, for each payment class, it includes equivalent BTC/USD value (using day-to-day average Bitcoin price).

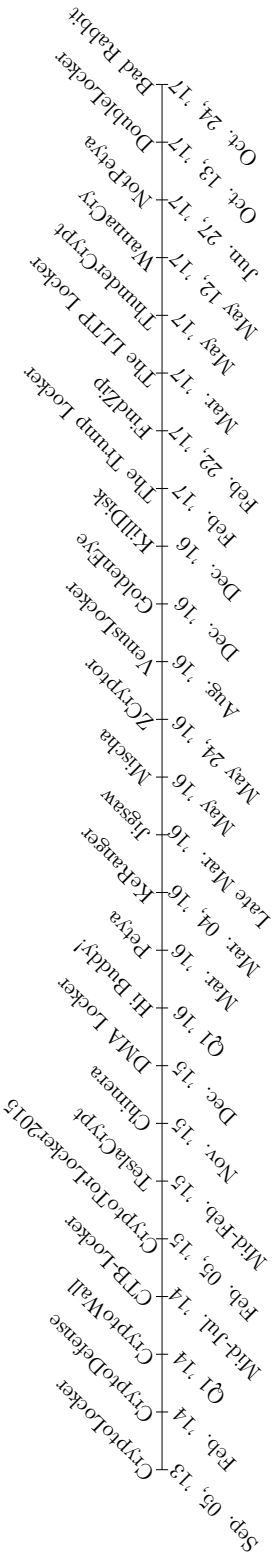


Figure 2.3: Occurrence of Bitcoin ransomware

It is clear that CryptoLocker received the maximum number of payments, i.e., 51,766 payments that worth 133,045.9961 BTC, which is approximately USD 42,292,191.17. However, our framework classified 3,730 payments received by CryptoWall as ransom payments, which is the maximum number of ransom payments extorted by any ransomware. These payments worth 5,351.2329 BTC or USD 2,220,909.12. On another side, KeRanger received the minimum number of overall payments as well as the ransom payments.

Ransomware	Overall			Ransom		
	Payments	BTC	USD Value	Payments	BTC	USD value
CryptoLocker	51,766	133,045.9961	42,292,191.17	804	1403.7548	449,274.97
CryptoDefense	128	138.3223	70,113.41	108	126.6960	63,859.49
CryptoWall	51,278	87,897.8510	45,370,589.00	3,730	5,351.2329	2,220,909.12
DMA Locker	298	1,433.3463	580,763.95	117	339.4591	178,162.77
NotPetya	70	4.1787	10,284.42	33	4.0576	9,835.86
KeRanger	13	10.0044	4,175.35	10	9.9990	4,173.12
WannaCry	341	53.2906	99,549.05	238	47.1743	86,076.76

Table 2.1: Summary of overall payments and ransom payments to the ransomware for which the observed payments align with their period of activity and ransom demands

Now, we discuss each ransomware in details.

2.4.1 CryptoLocker

Appeared in September 2013, CryptoLocker targets computers running Windows operating system. It uses “Microsoft Enhanced RSA and AES Cryptographic Provider (MS_ENH_RSA_AES_PROV)” to create encryption keys and to encrypt users’ files with the strong RSA (CALG_-RSA_KEYX) and AES (CALG_AES_256) algorithms. Before beginning the encryption process, it establishes a connection with its C&C to obtain an RSA public key. It encrypts each file with a unique AES key; after use, it encrypts each AES encryption key with the RSA public key [120].

Infection: CryptoLocker infection spread through two modes. In its initial release beginning from September 5, 2013, the cybercriminals especially targeted business professionals through spam emails. The messages of the emails were typical “customer complaints" against recipients’ firm. Attached to these emails was a ZIP archive that contained a single malicious Windows executable (exe) file. The names of both the ZIP file and malicious executable were identical (except for extensions) with 13 to 17 random alphabetical characters. Later versions of CryptoLocker, starting from October 7, 2013, were distributed by the peer-to-peer (P2P) Gameover ZeuS [193]. In this case, Gameover Zeus used Cutwail spam botnet to send a huge number of spam emails miming popular online retailers and banking institutions. These emails often contained spoofed order confirmations, invoices, or urgent mes-

sage for unpaid balances to entice victims to follow CryptoLocker exploit kits.

Ransom demand: The ransom note asks the victim to pay the ransom within 72 hours through any one of the various payment methods. It also threatens that not paying the ransom would lead to (allegedly) destruction of decryption keys. In the initial versions, the payment option included cashU, Ukash, paysafecard, Bitcoin, or MoneyPak. However, later the ransoms were collected only via Bitcoin or MoneyPak. All these payments methods are anonymous (or at least pseudo-anonymous), which makes it difficult to track the payer and the payee. The amount of demanded ransom and their corresponding timelines (both the dates are included) are as follow:

- 2 BTC between September 5, 2013 and November 11, 2013 allowing a three-day ransom period.
- 10 BTC between November 1, 2013 and November 11, 2013. The payment was the fee for using "CryptoLocker Decryption Service" that allowed victims, who failed to pay ransoms within the given time frame, to recover their files.
- 1 BTC between November 8, 2013 and November 13, 2013 to allowing a three-day ransom period.
- 0.5 BTC between November 10, 2013 and November 27, 2013 to allowing a three-day ransom period.
- 2 BTC between November 11, 2013 and January 31, 2014. In this case, the payment was the reduced fee for using "CryptoLocker Decryption Service".
- 0.3 BTC between November 24, 2013 and December 31, 2013.
- 0.6 BTC between December 20, 2013 and January 31, 2014.

Associated Bitcoin addresses and transactions: To evaluate the economic impact of CryptoLocker, we initially began with four Bitcoin addresses listed in Table C.1 in Appendix C. Using these addresses, Module 1 (Section 2.3.1) generated 956 addresses belonging to CryptoLocker cluster (C_{CL}). We obtained the detailed transaction history of these addresses using Module 2 (Section 2.3.2). Our analysis of transactions to C_{CL} reveals that C_{CL} received, in total, over 51,000 payments, which accounts for over 133,000 BTC (more than USD 42,000,000). Table 2.2 presents a summary of the total payments credited to C_{CL}.

Payments	BTC	USD valuez (daily highest BTC price)	USD value (daily average BTC price)	USD value (daily lowest BTC price)
51,766	133,045.9961	42,722,858.15	42,292,191.17	41,734,959.83

Table 2.2: Total payments credited to C_{CL} including all ransom and non-ransom payments

Economy of ransom payments in Bitcoin: To evaluate the gross economic impact of only the ransom payments, we filtered the transactions using: (i) the ransom amounts and their timeline, (ii) our classification criteria mentioned in Module 3 (Section 2.3.3). Figure 2.4 shows the total number of ransoms paid by the victims by date. C_{CL} received 33 payment on October 10, 2013, which is the maximum number of ransoms paid in a single day. However, as shown in Figure 2.5, C_{CL} received slightly more than 70 BTC on November 5, 2013, which is the maximum number of Bitcoin received in a single day. On another side, C_{CL} received slightly above USD 23,000 on November 8, 2013, which is the maximum USD collected in a single day, see Figure 2.5.

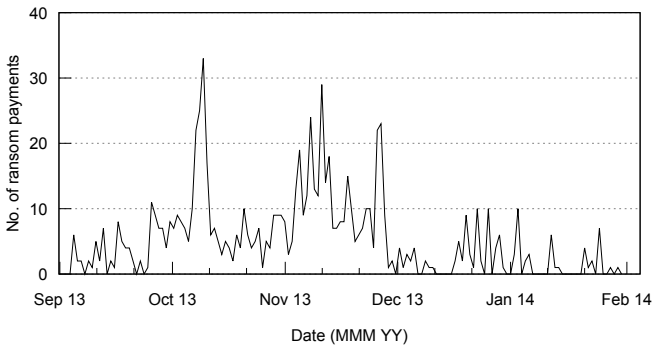


Figure 2.4: Number of ransoms paid to C_{CL}

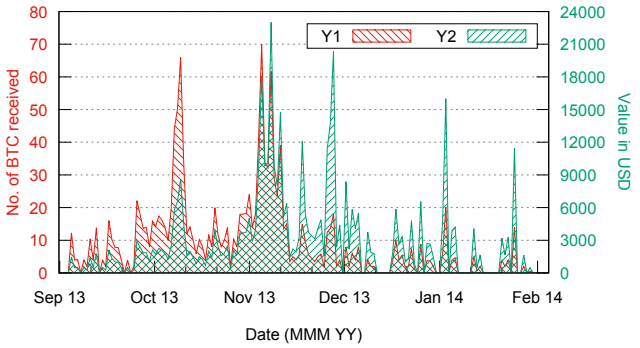


Figure 2.5: Number of Bitcoin received (in ransoms) by C_{CL} and their corresponding USD value

By further analyzing the addresses of C_{CL}, we discovered that approximately 83.16% Bitcoin addresses received maximum two payments. Moreover, 13.33% Bitcoin addresses received no more than one Bitcoin perhaps because victims were charged less due to a substantial increase in the Bitcoin value in late November 2013. Moreover, an address[7] collected 112.94 BTC while a different address[8] collected 83 ransom payments. These values correspond to the maximum number of Bitcoin and the maximum number of ransom collected by any address in C_{CL}. Figure 2.6 and Figure 2.7 depict Cumulative Distribution Function (CDF) of the number of ransoms and number of Bitcoin received (in ransoms) per address respectively.

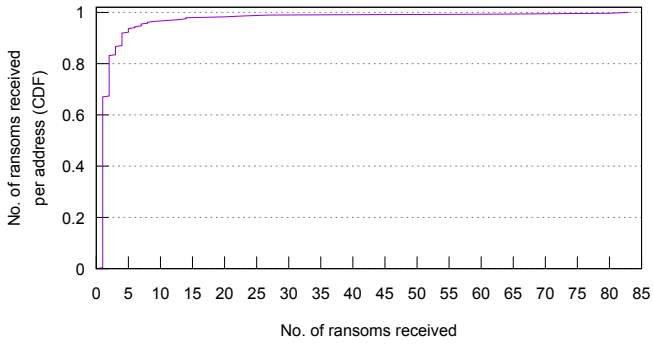


Figure 2.6: CDF of ransoms received per address in C_{CL}

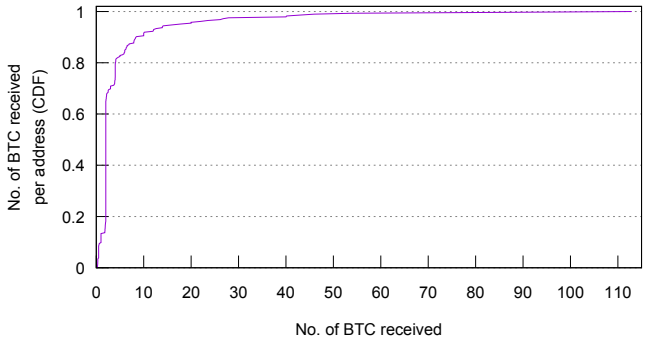


Figure 2.7: CDF of Bitcoin received (in ransoms) per address in C_{CL}

In total, we have identified 804 ransom payments to C_{CL}, which contribute to a total of 1,403.75 extorted BTC. Using day-to-day average Bitcoin price, we estimate that these ransoms convert to USD 449,274.97. Table 2.3 summarizes the ransoms paid to CryptoLocker.

[7] 16i7w5G2aoq8zqLDR3VJnawZ8VmYFZjVsd
[8] 1HFLn7JP7FZrufvNKkQPEfAWGjKUdFZEmy

Ransom	Time period	Payments	BTC	USD value
2 BTC	Sep. 05, '13 - Nov. 11, '13	443	884.9691	153,650.51
10 BTC (late)	Nov. 01, '13 - Nov. 11, '13	17	170.0000	47,549.90
1 BTC	Nov. 08, '13 - Nov. 13, '13	38	38.0000	14,302.26
0.5 BTC	Nov. 10, '13 - Nov. 27, '13	118	59.0000	37,108.27
2 BTC (late)	Nov. 11, '13 - Jan. 31, '14	106	212.0000	166,476.42
0.3 BTC	Nov. 24, '13 - Dec. 31, '13	31	9.1856	8,584.88
0.6 BTC	Dec. 20, '13 - Jan. 30, '14	51	30.6000	21,602.72
Total	**Sep. 05, '13 - Jan. 30, '14**	**804**	**1403.7548**	**449,274.97**

Table 2.3: Summary of ransoms paid to CryptoLocker

Although we cannot be sure that the unaccounted transactions are not ransom payments, our results align with the findings presented in works [120, 121, 138, 190] except for the final valuation in USD since the authors of these studies used the Bitcoin price on the day of their evaluation. More importantly, it implies that we can trust our methodology for evaluating other ransomware where a baseline for comparison is not available.

2.4.2 CryptoDefense

With a sophisticated hybrid design, CryptoDefense first appeared in the last week of February 2014. It incorporates many powerful techniques that were used by previous ransomware. For example, use of Bitcoin and the Tor network for anonymity, RSA-2048 based public-key cryptography for strong encryption, and the typical pressure tactics such as a short deadline for payment with threats of increasing the ransom after the deadline. It targets Windows systems. CryptoDefense encrypts files using the AES-256 algorithm. It generates the encryption key on the victim's computer using Windows CryptoAPI library. After the file encryption process completes, it encrypts the AES key using an RSA-2048 public key.

Infection: Primarily, CryptoDefense ransomware infiltrated via spam emails that contained malicious payload disguised as a compressed PDF document. Upon successful infiltration, it attempts to contact its C&C; and it sends information about the infected system in the initial communication. Upon receiving an acknowledgment from the C&C, it starts the encryption process.

Ransom demand: CryptoDefense asks USD/EUR 500 in Bitcoin within four days to decrypt the files. The cost of decryption after four days increases to USD/EUR 1,000. The attackers also provide a unique .onion page for each victim. Here, the victims could see a screenshot of their compromised system and decrypt one file as a proof of decryption.

Associated Bitcoin addresses and transactions: We began with two publicly known Bitcoin payment addresses of CryptoDefense. These addresses are listed in Table C.2 in Appendix C. In our analysis, the CryptoDefense

cluster (C_{CD}) had only two addresses as Module 1 generates no new address from these addresses. Our analysis of transactions (obtained using Module 2) to C_{CD} indicates that C_{CD} collected 128 payments. The total value of these payments is somewhat above 138 BTC (more than USD 70,000). Table 2.4 presents a summary of the total payments credited to C_{CD}.

Payments	**BTC**	**USD value (daily highest BTC price)**	**USD value (daily average BTC price)**	**USD value (daily lowest BTC price)**
128	138.3223	72,342.26	70,113.41	67,715.88

Table 2.4: Total payments credited to C_{CD} including all ransom and non-ransom payments

Economy of ransom payments in Bitcoin: Due to the limited number of transactions, we manually verified each payment to C_{CD}. As shown in Table 2.5, each Bitcoin address collected at minimum 35 ransom payments and a minimum of about 36.83 BTC.

Address	**Payments**	**BTC**
19DyWHtgLgDKgEeoKjfpCJJ9WU8SQ3gr27	35	36.8339
1EmLLj8peW292zR2VvumYPPa9wLcK4CPK1	73	89.8622

Table 2.5: Number of ransoms and Bitcoin received (in ransoms) per address in C_{CD}

Figure 2.8 shows the total number of ransoms paid, and Figure 2.9 depicts the corresponding number of Bitcoin received and their value in USD. Figure 2.8 and Figure 2.9 also depict that on March 28, 2014, C_{CD} collected around 13 BTC in 11 ransom payments, which amounts to approximately USD 6,500. It is the day when it received the maximum number of ransom payments/Bitcoin/USD in a single day.

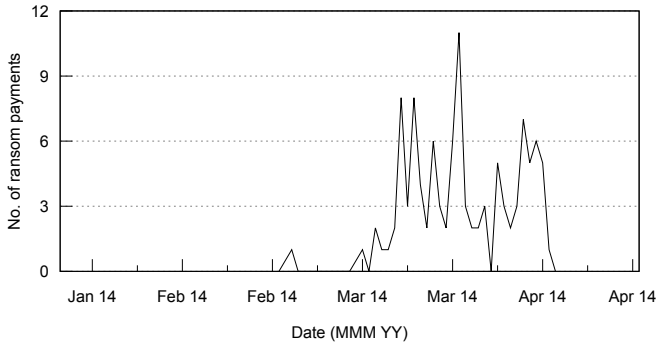


Figure 2.8: Number of ransoms paid to C_{CD}

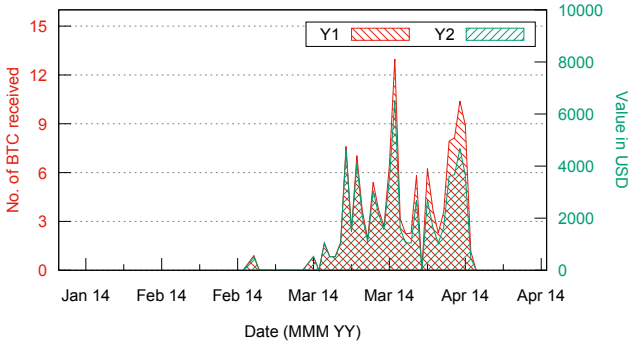


Figure 2.9: Number of Bitcoin received (in ransoms) by C_{CD} and their corresponding USD value

In total, we identified 108 ransom payments to C_{CD}, which corresponds to 126.70 extorted BTC. Using day-to-day average Bitcoin price, we compute that the value of these ransom payments is equivalent to USD 63,859.49. Table 2.6 summarizes the ransoms payments made to CryptoDefense.

Ransom	Time period	Payments	BTC	USD value
$/€500	Feb. 28, '14 - Apr. 11, '14	94	96.1758	49,271.63
$/€1,000		14	30.5202	14,587.86
Total		**108**	**126.6960**	**63,859.49**

Table 2.6: Summary of ransoms paid to CryptoDefense

Unexpectedly, CryptoDefense has a built-in flaw. It generates the asymmetric key pair on the victim's system. However, due to the poor implementation of the Microsoft's cryptographic infrastructure, it leaves a local copy of the keys. Anti-ransomware took advantage of this flaw to decrypt victim's computer. Such initiatives saved at least USD 175,000 worth ransoms [96].

2.4.3 CryptoWall

CryptoWall is recognized for its use of strong encryption algorithm, unique .CHM file infection mechanism, and strong C&C activity over the anonymous Tor network. According to the Dell SecureWorks Counter Threat Unit (CTU) research team [84], CryptoWall infection was spreading from the first half of November 2013. However, the attackers activated it in the first quarter of 2014. The earlier versions of CryptoWall closely impersonated both the appearance and the behavior of the CryptoLocker. It affects Windows operating systems by encrypting files using the RSA-2048 (and the AES-256 encryption algorithm from version 3.0) encryption algorithm.

Infection: Since its genesis, CryptoWall had spread through several infection vectors, which included drive-by downloads, browser exploit kits (e.g.,

Angler), and email attachments. Starting from late March 2014, it spread through download links sent via the Cutwail spam botnet and malicious email attachments. Interestingly, from June 2014, the malicious emails included links to popular cloud services such as Dropbox, MediaFire, and Cubby. The links pointed to a ZIP archive that contained the CryptoWall executable. Later these emails used a standard "missed fax" decoy and also mimicked message from government agencies or financial institutions that included links to malicious payload hosted over cloud services.

Evolution: Each version of CryptoWall lasted for a few months until a stealthier and enhanced version emerged.

- CryptoWall 1.0: Initial variants of CryptoWall lacked a unique name. It surfaced with its official name in the first quarter of 2014.
- CryptoWall 2.0: It appeared in November 2014. This version was almost identical to the previous version. However, unlike its predecessor, it creates a unique Bitcoin payment address for each victim and uses its own Web-2-Tor gateways.
- CryptoWall 3.0: The third version of CryptoWall emerged in January 2015. This version uses a local symmetric (AES-256) key for file encryption. The symmetric key is then encrypted using a unique public (RSA-2048) key generated by the C&C server. Such process of encryption is much faster as compared to the previous versions.
- CryptoWall 4.0: Another updated version with improved communications and better code design to exploit more vulnerabilities appeared in November 2015.

Ransom demand: The attackers originally accepted ransom payments through Litecoin [84]. However, the only witnessed Litecoin address[9] never collected any payment. Additionally, the victims could also pay the ransom via Bitcoin. The amount of ransom fluctuated frequently. Also, the time frame to pay the ransom varied up to seven days. According to our observation, the demanded ransom and their corresponding timelines (both the dates are included) are as follow:

- $200 worth BTC between March 2, 2014 and November 4, 2015.
- $500 worth BTC between March 2, 2014 and December 22, 2015.
- Late payment of $600 worth BTC between March 5, 2014 and November 5, 2015. This payment was three times the original ransom amount.

[9] LTv4m4y7NKHCXdw31dSEpTJmP6kXTinWDy

- Late payment of $1,000 worth BTC between March 5, 2014 and December 2, 2015. This payment was twice the original ransom amount.
- $700 worth BTC between March 10, 2014 and December 11, 2015.
- Late payment of $1,400 worth BTC between March 11, 2014 and December 21, 2015. This payment was twice the original ransom amount.

Associated Bitcoin addresses and transactions: We began with forty-two publicly known Bitcoin addresses of CryptoWall. These addresses are listed in Table C.3 in Appendix C. Using these addresses, Module 1 generated 2,944 addresses belonging to CryptoWall cluster (C_{CW}). Our analysis of transactions (obtained using Module 2) to C_{CW} shows that C_{CW}, in total, received over 51,000 payments. The total worth of these payments is nearly 88,000 BTC (more than USD 45,000,000). Table 2.7 presents a summary of the total payments credited to C_{CW}.

Payments	**BTC**	**USD value (daily highest BTC price)**	**USD value (daily average BTC price)**	**USD value (daily lowest BTC price)**
51,278	87,897.8510	46,526,673.59	45,370,589.00	44,020,263.63

Table 2.7: Total payments credited to C_{CW} including all ransom and non-ransom payments

Economy of ransom payments in Bitcoin: Using the timeline of ransom demands, we carefully analyzed all the transactions with Module 3 to distinguish ransom payments and evaluated the net worth generated by such payments. As shown in Figure 2.10 and Figure 2.11, on March 27, 2014, C_{CW} received slightly above 185 BTC in 158 payments. The total value of these payments is over USD 100,000. It is the day when it received the maximum number of ransom payments/Bitcoin/USD in a single day.

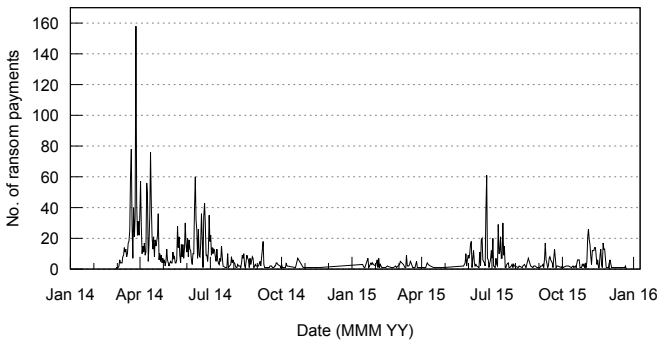


Figure 2.10: Number of ransoms paid to C_{CW}

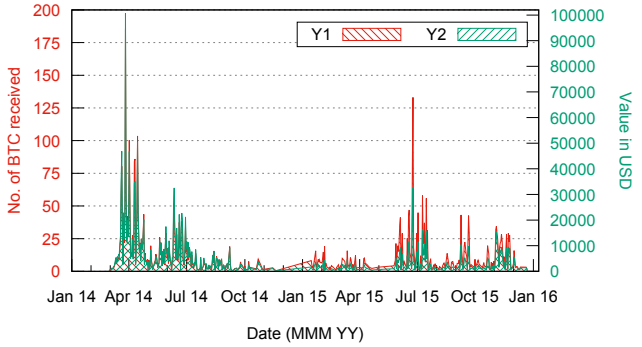


Figure 2.11: Number of Bitcoin received (in ransoms) by C_{CW} and their corresponding USD value

By investigating the addresses of C_{CW}, we observed that approximately 43.77% Bitcoin addresses received no more than one payment and 40.10% Bitcoin addresses collected maximum two Bitcoin. On another side, an address[10] collected 193.94 BTC in 209 ransom payments. These values correspond to the maximum number of Bitcoin and the maximum number of ransom collected by any address in C_{CW}.

We have identified 3,730 ransom payments to C_{CW}, which amount to 5,351.23 extorted BTC. Using day-to-day average Bitcoin price, we calculate that these ransom payments are equivalent to USD 2,220,909.12. Table 2.8 summarizes the ransoms paid to CryptoWall.

Ransom	Time period	Payments	BTC	USD value
$200	Mar. 02, '14 - Nov. 04, '15	614	232.3343	121,849.84
$500	Mar. 02, '14 - Dec. 22, '15	1,631	2220.9167	821,741.46
$600 (late)	Mar. 05, '14 - Nov. 05, '15	382	444.5144	226,558.14
$1,000 (late)	Mar. 05, '14 - Dec. 02, '15	423	836.5054	422,576.75
$700	Mar. 10, '14 - Dec. 11, '15	466	966.7365	327,518.98
$1,400 (late)	Mar. 11, '14 - Dec. 21, '15	214	650.2256	300,663.95
Total	**Mar. 02, '14 - Dec. 22, '15**	**3,730**	**5,351.2329**	**2,220,909.12**

Table 2.8: Summary of ransoms paid to CryptoWall

Moreover, according to the report by CTU researchers [84], CryptoWall attackers allowed the victims to decrypt their system by paying a further increased amount even after the expired deadline. Although, we have not directly observed any sample of CryptoWall demanding such compensations. Nevertheless, the timing and the volume of such payments suggest that these payments pertain to ransoms. Table 2.9 summarizes such payments.

[10] 17AGazRCLStNguMDCxDoj7ZQHvaZBWTJZj

Amount	Time period	Payments	BTC	USD value
$1,500	Mar. 12, '14 - Dec. 12, '15	222	678.7995	333,587.51
$1,750	Mar. 12, '14 - Nov. 04, '15	192	647.5063	336,578.87
$2,000	Mar. 06, '14 - Jul. 06, '14	170	650.7245	339,794.84
$10,000	Mar. 11, '14 - Jul. 11, '14	131	2623.3381	1,316,778.41
Total	**Mar. 06, '14 - Dec. 12, '15**	**715**	**4600.3684**	**2,326,739.63**

Table 2.9: Summary of high value (possibly ransom) payments to CryptoWall

If we add these payments to the originally identified ransom payments, then the revenue of CryptoWall reaches nearly 10,000 BTC, i.e., approximately USD 4,500,000.

2.4.4 DMA Locker

DMA Locker is one of the most actively developed and updated ransomware so far. From encryption algorithm to network communication, cybercrooks perpetually updated each component of DMA Locker. Initially, it used only the symmetric key cryptography for file encryption. However, later versions employ a stronger encryption approach by combining the AES-256 and the RSA-2048 encryption algorithms. It affects Windows operating system.

Infection: The distribution mechanism of DMA Locker also evolved with the course of time. The malicious payload was hosted on compromised websites, and their links were distributed via email spamming. It also infiltrated by hacking Remote Desktops. The latest edition of the ransomware also spread via Neutrino exploit kit [47].

Evolution: The development timeline of DMA Locker is discussed below:

- DMA Locker 1.0: The first version of DMA Locker was noticed in the last week of December 2015 with support for two languages: Polish and English. It performs file encryption by using the AES-256 algorithm in ECB mode. It uses a single AES key to encrypt target files, which is stored in the binary and deleted after use.
- DMA Locker 2.0: On February 3, 2016, DMA Locker was updated to use separate keys for each file. After encrypting a file, it encrypts the used AES key by a hardcoded RSA public key and stores the encrypted AES key in the encrypted file.
- DMA Locker 3.0: Due to weak implementation of the random number generator, the AES key generated by the previous version can be guessed. In view to fix the flaw, the third edition was released on February 22, 2016. However, the entire campaign used the same RSA key-pair. Meaning that single private key can be reused for decrypting other infected systems.

- DMA Locker 4.0: The latest version of DMA Locker was released on May 19, 2016. This version generates a unique RSA key-pair on the server for each victim. Unlike previous versions, DMA Locker 4.0 can not work offline because it is designed to download the asymmetric public key from the server [144].

Ransom demand: The cybercrooks behind DMA Locker accepted ransom payments through Bitcoin. DMA Locker 4.0 gives payment instructions on a website. The website was a regularly (not Tor-based) hosted site. Surprisingly, the payment site used the same IP address as the C&C. Similar to other components, the ransom amount was also updated with time. Moreover, the first two versions stipulate a strict deadline of four days to pay the ransom. Other versions allow an extension of three days at the cost of an increased ransom. The demanded ransom and their corresponding timelines (both the dates are included) are as follow:

- 1 BTC between December 28, 2015 and July 22, 2016.
- 1.3 BTC between January 19, 2016 and May 30, 2016.
- 2 BTC between January 28, 2016 and July 22, 2016 to allowing a three-day ransom period.
- 4 BTC between February 22, 2016 and June 5, 2016 to allowing a three-day ransom period.
- 8 BTC as late fee between February 22, 2016 and August 5, 2016.
- 1.5 BTC as late fee between May 19, 2016 and July 11, 2016.
- 3 BTC between May 24, 2016 and August 25, 2016.

Associated Bitcoin addresses and transactions: To understand the economic impact of DMA Locker, we began with eight Bitcoin addresses listed in Table C.4 in Appendix C. Using these addresses, Module 1 generated 28 addresses belonging to DMA Locker cluster (C_{DL}). Our scrutiny of transactions (obtained using Module 2) to C_{DL} shows that C_{DL} received altogether 298 payments, i.e., more than 1,400 BTC (over USD 580,000). Table 2.10 presents a summary of the total payments credited to C_{DL}.

Payments	**BTC**	**USD value (daily highest BTC price)**	**USD value (daily average BTC price)**	**USD value (daily lowest BTC price)**
298	1,433.3463	593,498.26	580,763.95	567,543.86

Table 2.10: Total payments credited to C_{DL} including all ransom and non-ransom payments

Economy of ransom payments in Bitcoin: We used Module 3, guided by the timeline of ransom demands, to separate ransom payments. Figure 2.12 depicts the total number of ransoms paid by date. C_{DL} received 5 payment on April 27, 2016, which is the maximum number of ransoms paid in a single day. On another side, as shown in Figure 2.13, C_{DL} collected 12 BTC on May 19, 2016, which corresponds to the maximum number of Bitcoin received in a single day. Furthermore, C_{DL} received over USD 6,300 on August 5, 2016, which stands for the maximum USD received in a single day, see Figure 2.13.

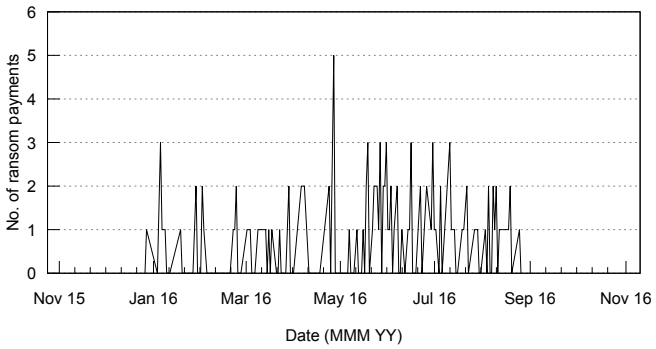


Figure 2.12: Number of ransoms paid to C_{DL}

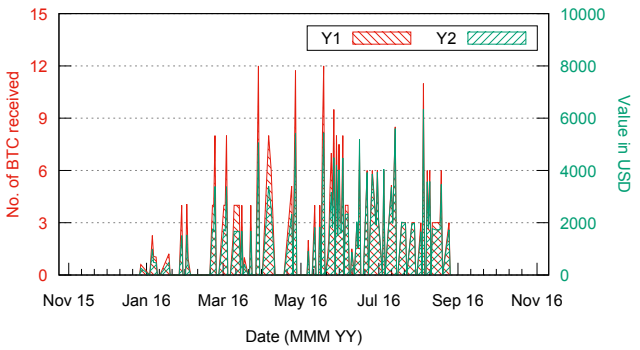


Figure 2.13: Number of Bitcoin received (in ransoms) by C_{DL} and their corresponding USD value

We further found that around 30% addresses in C_{DL} collected no more than one payment and nearly 20% Bitcoin addresses received less than one Bitcoin. Furthermore, an address[11] collected 112.87 BTC in 38 ransom payments. These values correspond to the maximum number of Bitcoin and the

[11] 1LPgKoErPUeM92SDY5axJzYCdQbeiRHD6i

maximum number of ransom collected by any address in C_{DL}. Table 2.11 summarizes the ransoms paid to DMA Locker.

Ransom	Time period	Payments	BTC	USD value
1 BTC	Dec. 28, '15 - Jul. 22, '16	16	14.7526	7,052.37
1.3 BTC	Jan. 19, '16 - May 30, '16	4	5.2470	2,424.01
2 BTC	Jan. 28, '16 - Jul. 22, '16	16	32.0809	16,638.46
4 BTC	Feb. 22, '16 - Jun. 05, '16	33	131.9950	60,443.98
8 BTC (late)	Feb. 22, '16 - Aug. 05, '16	4	32.4892	16,960.59
1.5 BTC (late)	May 19, '16 - Jul. 11, '16	6	8.9147	5,136.87
3 BTC	May 24, '16 - Aug. 25, '16	38	113.9797	69,506.49
Total	**Dec. 28, '15 - Aug. 25, '16**	**117**	**339.4591**	**178,162.77**

Table 2.11: Summary of ransoms paid to DMA Locker

We have identified 117 ransom payments to C_{DL}, which contribute to a total of 339.46 extorted BTC. Using day-to-day average Bitcoin price, we estimate that these ransom payments value USD 178,162.77.

2.4.5 Petya

Initially seen in March 2016, this family of malware denies access to the full system by targeting the low-level structures on the disk. Petya spread via emails, and was delivered as Windows executable with an icon of a PDF document. Upon running, it opens a User Account Control (UAC) window. Accepting UAC allows Petya to run. In this case, it overwrites the Master Boot Record (MBR) with a custom bootloader that loads a malicious kernel. Then, this kernel encrypts the Master File Table (MFT) using Salsa20 stream cipher with a 32-byte long key, which leaves file system unreadable. Figure 2.14 depicts the full process of Petya.

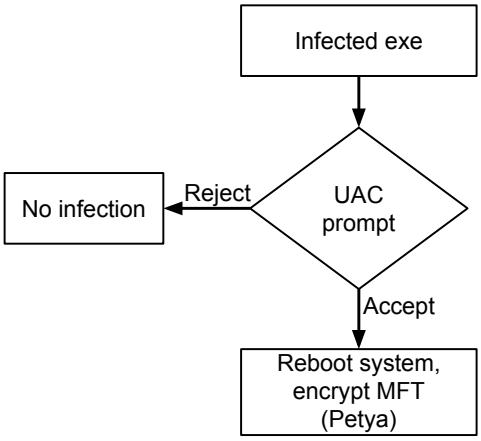


Figure 2.14: Workflow of Petya

Mischa: In May 2016, the malware was modified to integrate another malicious payload known as Mischa. Mischa was designed as a backup strategy to Petya. Altogether, they target different (both high-level and low-level) layers of a system. In this version, denying the UAC prompt directs Mischa to encrypt local files on the victim computer; otherwise, Petya proceeds. Figure 2.15 depicts the full process of Mischa. Both Petya and Mischa can work offline without communicating with their C&C. The payload from the dropper[12] uses CryptGenRandom function from the Windows CryptoAPI library to generate a random encryption key. Mischa uses a CBC-style file encryption utilizing a randomly generated key along with the previously generated master key. Interestingly, Mischa can encrypt documents as well as executables [38]. The cybercriminals also offered RaaS through their own affiliate program.

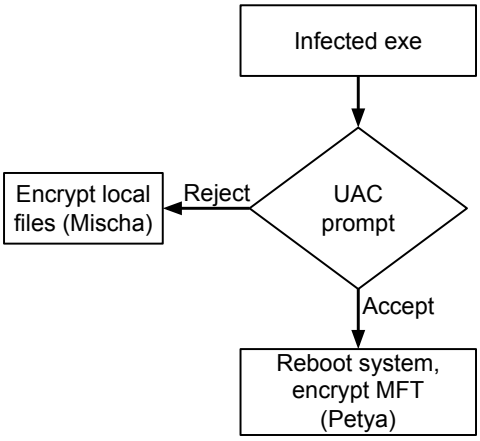


Figure 2.15: Workflow of Mischa

GoldenEye: The malware was again rebranded as GoldenEye in early December 2016. In contrast with the previous versions, GoldenEye executes both payloads, where possible. Similar to its predecessors, it was also distributed via email. But, the payload was attached to an MS Excel document. The document prompts the user to enable Macro content. Enabling Macro content executes a malicious Visual Basic Script, which runs the Mischa payload to encrypt documents on the system. After Mischa finishes, it attempts to gain system privileges via DLL injection (Windows 7 - 8.1), or a UAC prompt is shown (Windows 10). If DLL infection succeeds or the UAC prompt is accepted, Petya payload encrypts the MFT. Figure 2.16 depicts the full process of GoldenEye.

[12]The file that launches a malware.

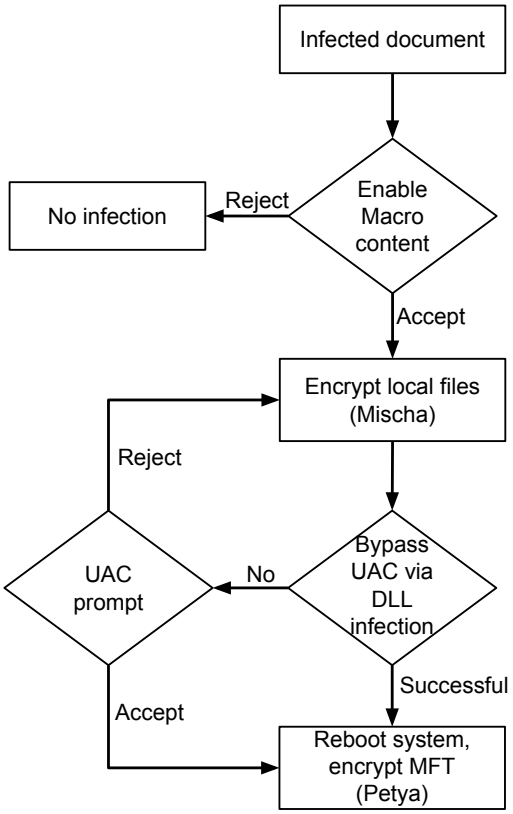


Figure 2.16: Workflow of GoldenEye

NotPetya: The latest variant of Petya surfaced on June 27, 2017. Kaspersky unofficially named[13] it NotPetya/ExPetr due to significant differences in its operations compared to the earlier versions. Initially, NotPetya was distributed as an update to MeDoc[14] accounting software prevalent in Ukraine. After infiltration, it self-propagates via two methods. One of the methods is the EternalBlue exploit, which is an exploit of Windows' Server Message Block (SMB) protocol. The same exploit is also used by WannaCry ransomware, which was released only a month before NotPetya. It can also spread across network shares by Windows Management Instrumentation Commandline (WMIC), for which it uses credentials acquired from the local machine. In contrast with other ransomware, it focuses on the local network to spread rather than the Internet. NotPetya works as a destructive data wiper tool due to its inability to restore the encrypted sectors of the physical disk [197].

Associated Bitcoin addresses and transactions: We discuss the financial

[13]https://tinyurl.com/y4sqjo8o
[14]https://tinyurl.com/y2sy6oy7

transactions associated with only NotPetya because the payments received by the address clusters generated for Mischa and GoldenEye (using addresses listed in Table C.5 and Table C.6 in Appendix C, respectively) were significantly less (no more than USD 3) than the demanded ransoms (roughly USD 1,000). For NotPetya, cybercriminals used a single Bitcoin payment address to collect a fixed ransom of USD 300. The address is listed in Table C.7 in Appendix C. NotPetya cluster (C_{NP}) generated by Module 1 also had only one Bitcoin addresses. We acquired the detailed transaction history of this address using Module 2. C_{NP} received exactly 70 payments. These payments worth slightly above 4 BTC (over USD 10,000). Table 2.12 summarizes the payments credited in C_{NP}.

Payments	**BTC**	**USD value (daily highest BTC price)**	**USD value (daily average BTC price)**	**USD value (daily lowest BTC price)**
70	4.1787	10,717.74	10,284.42	9,958.33

Table 2.12: Total payments credited to C_{NP} including all ransom and non-ransom payments

Economy of ransom payments in Bitcoin: We segregated ransom payments using Module 3. As shown in Figure 2.17 and Figure 2.18, on the day of its outbreak, i.e., on June 27, 2017 C_{NP} received somewhat above 3 BTC in total 27 payments that amount approximately USD 8,000. It collected the maximum number of ransom payments/Bitcoin/USD on this day.

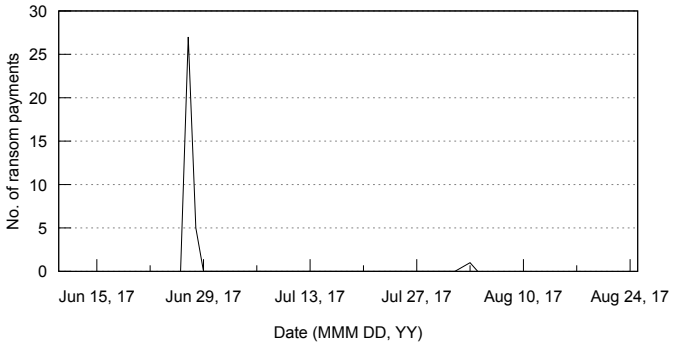


Figure 2.17: Number of ransoms paid to C_{NP}

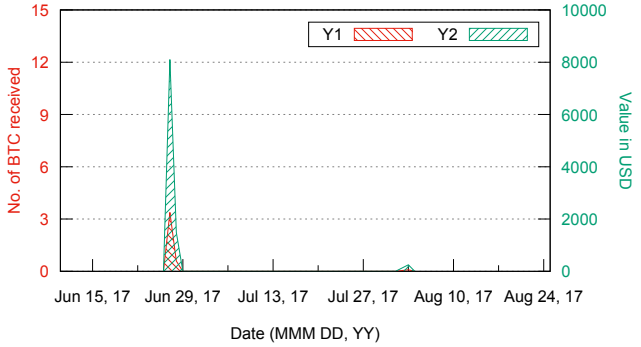


Figure 2.18: Number of Bitcoin received (in ransoms) by C_{NP} and their corresponding USD value

In total, we have identified 33 ransom payments to C_{NP}, which add up to roughly 4.06 extorted BTC. Using day-to-day average Bitcoin price, we calculate that these ransom payments worth equivalent to USD 9,835.86. Table 2.13 summarizes the ransoms paid to NotPetya.

Ransom	Time period	Payments	BTC	USD value
$300	Jun. 27, '17 - Aug. 03, '17	33	4.0576	9,835.86

Table 2.13: Summary of ransoms paid to NotPetya

Given the irreversible destructive nature and the targeted-software of NotPetya, many researchers suggested that the primary aim of NotPetya was not money. Other researchers speculated that it was probably a second level attack to wipe traces of an early intrusion [141, 184].

2.4.6 KeRanger

KeRanger emerged as the first fully functional ransomware that targets macOS operating system. It was discovered on March 4, 2016, by Palo Alto Networks. By nature it is a trojan horse, it uploads infected system's information (e.g., model name, UUID) to its C&C over the Tor network to obtain an RSA public key. Along with the key it also receives victim-specific information that it is writes to a file named "README_FOR_DECRYPT.txt." KeRanger encrypts each file F as follows:

1. Generate a random number (R).
2. Generate an Initialization Vector (I) using F's content.
3. Encrypt R with the RSA key (obtained from C&C), and store it at the beginning of *F.encrypted* file.

4. Store I inside the *F.encrypted* file.

5. Mix R and I to generate an AES key.

6. Encrypt data of the original file with the AES key and write the encrypted data to *F.encrypted* file [50].

Infection: KeRanger was disseminated via two infected installers for the open source BitTorrent client project Transmission version 2.90, which were available for download on the official website. Moreover, these installers were signed with a valid Mac app development certificate; hence, they bypassed OS X's Gatekeeper security feature.

Ransom demand: To decrypt the encrypted files, the cybercrooks asked the victims to pay exactly one Bitcoin (around USD 400) through a website hosted on the Tor network.

Associated Bitcoin addresses and transactions: We began with six identified Bitcoin address of KeRanger. These addresses are listed in Table C.8 in Appendix C. Module 1 identified ten new addresses from these six addresses. Therefore, KeRanger cluster (C_{KR}) had a total of 16 addresses in our analysis. The transactions (obtained using Module 2) to C_{KR} show that C_{KR}, in total, received only 13 payments. These transactions worth around 10 BTC (nearly USD 4,200). Table 2.14 presents a summary of the total payments credited to C_{KR}

Payments	**BTC**	**USD value (daily highest BTC price)**	**USD value (daily average BTC price)**	**USD value (daily lowest BTC price)**
13	10.0044	4,204.54	4,175.35	4,147.01

Table 2.14: Total payments credited to C_{KR} including all ransom and non-ransom payments

Economy of ransom payments in Bitcoin: We isolated ransom payments using Module 3. Figure 2.19 shows the total number of ransoms paid to C_{KR}. C_{KR} received the last ransom payment on March 17, 2016. Figure 2.20 depicts the total number of Bitcoin received (in ransom) and their corresponding value in USD. Moreover, we found that none of the address received more than one Bitcoin (more than one ransom, in other words).

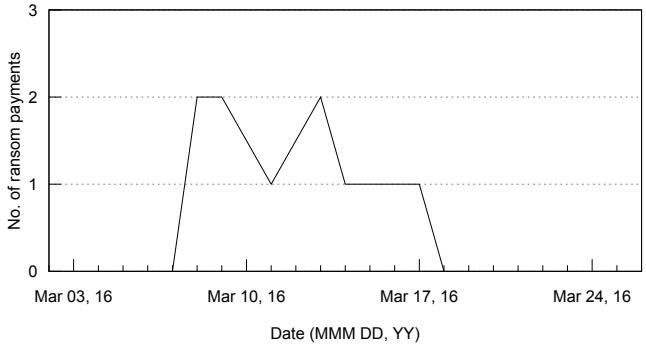


Figure 2.19: Number of ransoms paid to C_{KR}

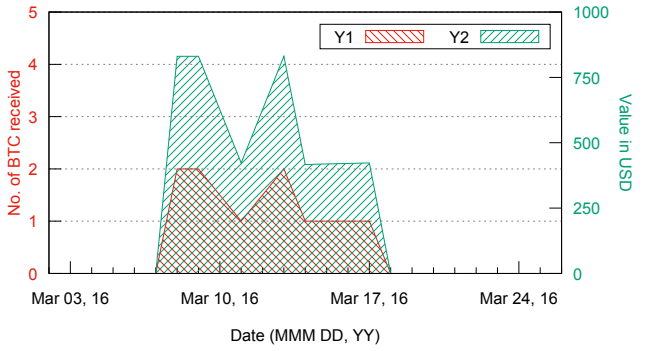


Figure 2.20: Number of Bitcoin received (in ransoms) by C_{KR} and their corresponding USD value

According to our analysis, C_{KR} received only 10 ransom payments, which contribute to roughly 9.99 extorted BTC. Using day-to-day average Bitcoin price, we estimate that these ransoms convert to USD 4,173.12. Table 2.15 summarizes the ransoms paid to KeRanger.

Ransom	Time period	Payments	BTC	USD value
1 BTC	Mar. 04, '16 - Mar. 17, '16	10	9.9990	4,173.12

Table 2.15: Summary of ransoms paid to KeRanger

One of the possible reasons for such low number of ransom payments could be that by March 5, 2016, Transmission project removed the infected installers from the website, and Apple revoked the abused certificate that allowed Gatekeeper to block the infected installers.

2.4.7 WannaCry

WannaCry (also known as WCry, WanaCrypt0r, Wana Decrypt0r 2.0) is blended threat with characteristics of both a worm and a ransomware. It was first seen on May 12, 2017. It affects Windows system by encrypting files using a combination of the RSA and the AES algorithms. Interestingly, it encrypts each file with a separate 128-bit AES encryption key in CBC mode. Furthermore, it encrypts each AES key individually using the RSA-2048 encryption algorithm [65].

Infection: WannaCry scans explicitly for the presence of the DoublePulsar backdoor on a target. If the DoublePulsar backdoor is not present, then it tries to compromise the system using the EternalBlue exploit [143]. The EternalBlue exploit was exposed merely a few months before the WannaCry attack by a hacker group known as The Shadow Brokers.

Kill switch and kill mutex: A kill switch is usually employed to terminate a program's execution. In case of WannaCry, the kill switch was a domain name[15]. Upon initialization, WannaCry tries to connect to the domain over HTTP. If the connection is successful, then it stops and exits. Possibly, it was designed to evade a sandbox testing. The kill switch domain was hardcoded in the source code and was discovered by Marcus Hutchins[16]. On another side, before beginning the encryption process, WannaCry attempts to create a mutex named "MsWinZonesCacheCounterMutexA" and exits if the mutex is already present.

Ransom demand: The ransom note asks the victims to pay USD 300 ransom in Bitcoin within three days. The ransom note also states that the ransom amount would become double (i.e., USD 600) after three days, and if the ransom is not paid within seven days from the day of infection, all the encrypted files would be deleted.

Associated Bitcoin addresses and transactions: Cybercriminals intended to create a unique Bitcoin payment address for each victim. But a race condition bug prevents the correct execution of the code. In this situation, it presents one of three hard-coded Bitcoin addresses to collect the ransom [198]. These addresses are listed in Table C.9 in Appendix C. Moreover, using these addresses, Module 1 generated no new address. Hence, WannaCry cluster (C_{WC}) generated by our framework had only three Bitcoin addresses during our analysis. We procured the detailed transaction history of these three addresses using Module 2. C_{WC} received 341 payments. These

[15]https://tinyurl.com/n5fy8nk
[16]https://tinyurl.com/y4t9gv39

payments worth over 50 BTC (approximately USD 100,000). Table 2.16 summarizes the payments credited in C_{WC}.

Payments	BTC	USD value (daily highest BTC price)	USD value (daily average BTC price)	USD value (daily lowest BTC price)
341	53.2906	102,141.19	99,549.05	96,497.20

Table 2.16: Total payments credited to C_{WC} including all ransom and non-ransom payments

Economy of ransom payments in Bitcoin: Due to comparatively a smaller number of transactions, we manually verified each payment to C_{WC}. As shown in Table 2.17, each Bitcoin address collected at minimum 69 ransom payments and a minimum of nearly 13.52 BTC.

Address	Payments	BTC
12t9YDPgwueZ9NyMgw519p7AA8isjr6SMw	77	15.1129
13AM4VW2dhxYgXeQepoHkHSQuy6NgaEb94	92	18.5431
115p7UMMngoj1pMvkpHijcRdfJNXj6LrLn	69	13.5183

Table 2.17: Number of ransoms and Bitcoin received (in ransoms) per address in C_{WC}

Figure 2.21 and Figure 2.22 indicate that on May 15, 2017, C_{WC} received 70 payments that amount to nearly 14 BTC, which is approximately USD 24,000. It is the day when it received the maximum number of ransom payments/Bitcoin/USD in a single day.

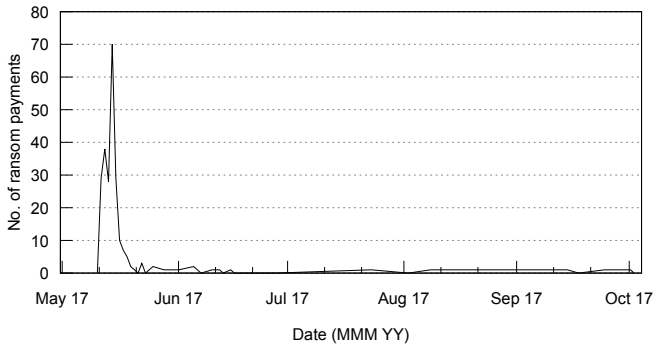


Figure 2.21: Number of ransoms paid to C_{WC}

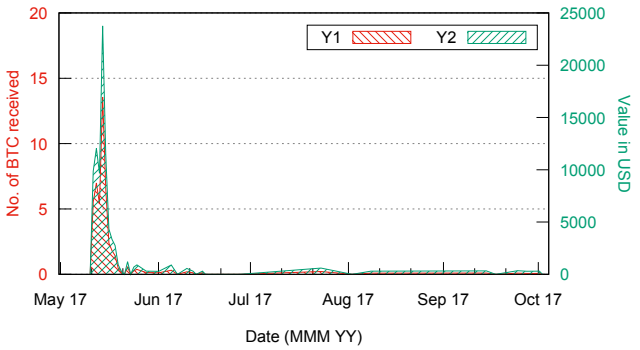


Figure 2.22: Number of Bitcoin received (in ransoms) by C_{WC} and their corresponding USD value

In total, we have identified 238 ransom payments to C_{WC}, which add up to 47.17 extorted BTC. Using day-to-day average Bitcoin price, we calculate that these ransom payments worth equivalent to USD 86,076.76. Table 2.18 summarizes the ransoms payments made to WannaCry.

Ransom	Time period	Payments	BTC	USD value
$300	May 12, '17 - Oct. 02, '17	192	32.3430	58,416.62
$600		46	14.8313	27,660.14
Total		**238**	**47.1743**	**86,076.76**

Table 2.18: Summary of ransoms paid to WannaCry

The overall impact (including financial losses) due to WannaCry infection could have been worse. Nevertheless, thanks to the early detection of the kill switch, which prevented the infected computers from spreading WannaCry further.

2.5 Limitations

One of the most important and decisive elements for the quality of the outcomes of our framework is the address identification module, presented in Section 2.3.1. It relies on the Bitcoin addresses collected from the public sources; the quality of data collected from the public sources could be a concern. One of the promising alternatives is to collect binaries of the ransomware and execute them several times in a virtual environment to witness/obtain Bitcoin addresses. However, the question of integrity and authenticity of the binaries remains the same. Given the nature of the problem, we followed the approach used in the previous studies [138, 190] and took extreme precaution while collecting addresses from the public sources.

The fundamental principles of the Bitcoin protocol implicitly impart two types of flaws in our address identification module: overestimation and un-

derestimation. Our methodology would overestimate when multiple users pool their transactions into a single transaction; as in the case of mixers. On another side, it would underestimate when there exists no evidence (in the blockchain) of an address owned by a user being used in conjunction with any other address of the same user. However, in a given scenario, it would report more accurate results as compared to the existing approaches due to its attributes of ransom classifications.

2.6 Summary

Pseudo-anonymity and irreversibility of Bitcoin transaction protocol have made Bitcoin a dexterous utility among cybercriminals. Unlike genuine users, who seek to transact securely and efficiently; cybercrooks exploit these characteristics to commit immutable and presumably untraceable monetary fraud. In this chapter, we have presented our comprehensive and longitudinal study on recent Bitcoin ransomware along with their renamed/rebranded versions. We have also introduced our framework to identify, collect, and analyze Bitcoin addresses that belong to the cybercriminals behind the ransomware. Moreover, we elaborated the characteristics and functionality of the ransomware as well as reported the economic impact of such ransomware from the Bitcoin payment perspective.

Chapter 3

Privacy Issues in Bitcoin Wallet Apps

Smartphones have become an integral part of our daily lives. Global smartphone sales have observed an enormous 1200% increase [160] in the past decade. Generally, smartphones are equipped with standard utilities and tools out of the box. To integrate supplementary features, the users can download and install additional applications, also called apps.

Bitcoin has gained enormous attention when its price raised from approximately $1,000 in March 2017 to nearly $19,500 in December 2017 [59]. The market capitalization of Bitcoin also grew dramatically and reached a level of $326 billion in Q4 2017. This growth in the value of Bitcoin attracted masses to trade this cryptocurrency, and platforms such as Coinbase enabled users to trade it conveniently. Moreover, several online shopping websites, supermarkets, etc. accept Bitcoin as a valid mode of payment. Consequently, an increasing number of users are adopting smartphone-based Bitcoin wallet apps for different purposes such as payment, remittance, and trading.

Network traffic analysis, on the other side, has been extensively exploited by adversaries to create user profiles and deduce sensitive information such as which airline does the user fly with, which financial institution does the user banks with, or which company provides insurance to the user. User profiling is also used for other non-malicious activities such as network performance optimization in the culture of bring your own device [158]. For traffic analysis, traditional TCP/IP traffic may be identified using port information because applications tend to use "well-known" port numbers that are typically reserved for standard services. Furthermore, to distinguish multiple sources of data-traffic from services that utilize the same port number (e.g., Internet browsing), it may sometimes adequate to inspect the HTTP headers, in particular, IP addresses to identify the communicating peers. However,

traffic analysis in the domain of the smartphones is complex because several apps exchange data using HTTP/HTTPS. If the developers choose to use HTTPS, the communication is encrypted, and thus, conventional techniques of inspecting the traffic cannot help in the task of traffic identification. Moreover, many apps as well as ad networks use Content Distribution Network (CDN) for scalable delivery of content and furnish Application Programming Interfaces (APIs) to their applications. Using CDNs and APIs may cause different apps to communicate via the same IP address (or IP range), which consequently hinders the identification techniques that solely depend on the IP addresses.

In this chapter, we focus on identifying activities of smartphone-based Bitcoin wallet users via network traffic analysis. We motivate our work by outlining one of the most critical situations arising in today's cyber-space: pseudo-anonymity offered by the Bitcoin system makes it difficult for law-enforcing agencies to trace the masterminds behind modern cybercrimes, e.g., ransomware campaigns, which are increasing day-by-day. We believe that our work can assist in the hunt of cybercriminals by monitoring (or at least by filtering) the potential Bitcoin wallet users. Most importantly, our work can be extended to other categories of smartphone apps to improve user profiling even further.

Contributions: We present our approach to identify user activities on smartphone-based Bitcoin wallet apps. To accomplish our goal, we did network traffic analysis using machine learning techniques. Since we focus on apps of the same type/functionality, it makes our classification problem even more difficult compared to classifying apps tailored for discrete purposes. Moreover, our goal is to identify user activities even in the presence of encryption.

Organization: The remainder of this chapter is organized as follows: Section 3.1 discusses the previous works on traffic analysis. Section 3.2 elaborates our system's design. Section 3.3 covers the details of our classifier. We present and discuss our results in Section 3.4. Finally, Section 3.5 summarizes the chapter.

3.1 Related Works

Since our work relies on network traffic analysis, we will primarily discuss the previous works related to it. Network traffic analysis using machine learning techniques has been an active area of research. Several attempts have been made to analyze network traffic from workstations, smartphones, etc. On the surface, smartphones' traffic analysis may appear as a sheer translation of existing works for workstations. However, several studies [101, 113, 136, 215]

have concluded that despite having similarities (e.g., end-to-end communication using via IP addresses and ports), there are nuances in the characteristics of the traffic generated by smartphones. Here, we will discuss the works related only to the smartphones; i.e., the main focus of our work. However, the interested readers may refer to the works [51, 114, 115, 139, 164, 174] to comprehend traffic analysis on workstations.

In the domain of the smartphones, network traffic analysis has been effectively utilized to leak sensitive user data [97], to find device's location [37], and to profile users based on the apps installed on their device [192]. Dai et al. [82] propose NetworkProfiler to automate profiling and identification of Android apps. It scrutinizes HTTP payload, and thus the approach is not adequate when the payload is encrypted. Wang et al. [207] present an approach for inspecting encrypted 802.11 frames to identify apps from App Store. Qazi et al. [169] propose a framework, called Atlas, to recognize Android apps using network flows obtained by leveraging SDN's data reporting. Mongkolluksamee et al. [153] use communication patterns and packet size distribution to distinguish among distinct Android apps. Alan and Kaur [36] use TCP/IP headers of the first 64 packets generated upon app launch to identify Android apps. Taylor et al. [201] showed that a passive eavesdropper can recognize Android apps by fingerprinting network traffic. Conti et al. [62] suggest eavesdropping encrypted network traffic to classify user actions within the scope of different Android apps. Similarly, Saltaformaggio et al. [182] propose NetScope, which also examines encrypted traffic to identify user activity. The work presented in [64] focuses on iMessage and three other third-party messaging apps for iOS to detect messaging activity. Zhou et al. [217] target a specific user action (i.e., send a tweet) on the Twitter app installed on an Android smartphone.

To summarize, existing solutions either focus on apps from different categories that inherently generate distinct network traffic or consider a particular smartphone platform. Our work is different from the state-of-the-art on two dimensions: (1) we focus on apps of same type/functionality, which makes classification problem even more difficult; (2) we consider both Android and iOS operating system, which collectively covers the majority of smartphone users.

3.2 System Design

We elaborate our decision for selecting the smartphones, apps, and their actions in Section 3.2.1. We also elucidate our equipment setup used for our experiments in Section 3.2.2.

3.2.1 Smartphone, App, and Action Selection

According to the report by Gartner [105], Android and iOS devices together accounted for 99.9% of all smartphone sales by the end of the year 2017. Hence, in our study, we used both Android (Samsung Galaxy S5 running Android 6.0.1) and iOS (iPhone 5 running iOS 10.3.3) smartphones. Table 3.1 lists the worldwide most downloaded [45] Bitcoin wallet apps on both Google Play Store and Apple's App Store in the year 2017.

N.	Google Play Store	App Store
1	Coinbase	Coinbase
2	Zebpay	Blockchain
3	Bitcoin Wallet (Bitcoin.com)	Bread
4	Luno	Bitcoin Wallet (Bitcoin.com)
5	Xapo	Xapo
6	Unocoin	BitPay
7	Mycelium	Zebpay
8	Wirex	Wirex
9	Bitcoin Wallet (Bitcoin Wallet Devs)	BTC.com
10	BTC.com	Copay

Table 3.1: Top 10 most downloaded Bitcoin wallet apps

From the apps listed in Table 3.1, we considered nine apps. These apps are listed in Table 3.2. The apps we omitted in our experiments are either country restricted (e.g., Unocoin requires Indian phone number and tax code to register) or available for both the platforms with identical features (e.g., Wirex). However, as the representative of the later class of apps, we included Bitcoin Wallet (Bitcoin.com) app for both the platforms. For non-Bitcoin apps, we chose the top-10 apps along with additional 20 Internet-dependent apps from the respective official application store of each platform. It is important to mention that these numbers do not include system apps and the apps that do not require the Internet, e.g., calculator app.

We inspected each app and identified the actions available on it. For Bitcoin wallet apps, we found seven classes of actions relevant to Bitcoin transactions: open the app, receive Bitcoin, send Bitcoin, generate a new Bitcoin address, buy/sell (trade) Bitcoin, see transaction history, and check available balance. We omitted other actions available on the wallet apps because they do not necessarily elicit network traffic, e.g., exit/close the app

	Android	iOS
Wallet apps	BTC.com	BitPay
	Bitcoin Wallet (Bitcoin.com)	
	Coinbase	Blockchain
	Luno	Bread
	Mycelium	Copay
Non wallet apps	Top-10 apps along with additional 20 Internet-dependent apps.	

Table 3.2: Apps used for classification

and share the app via SMS/Bluetooth. Table 3.3 lists the actions available on the wallet apps mentioned in Table 3.1.

App	Action						
	Open app	Receive Bitcoin	Send Bitcoin	Generate addresses	In-app buy/sell	Transaction history	Check balance
BTC.com	✓	✓	✓	✗	✠	❋	❋
BitPay	✓	✓	✓	✓	✓	♣	❋
Bitcoin Wallet (Bitcoin Wallet Devs)	✓	✓	✓	✗	✗	❋	❋
Bitcoin Wallet (Bitcoin.com)	✓	✓	✓	✓	✠	♣	❋
Blockchain	✓	✓	✓	✗	✓	♣	❋
Bread	✓	✓	✓	✗	✓	♣	❋
Coinbase	✓	✓	✓	✗	✓	♣	❋
Copay	✓	✓	✓	✓	✓	♣	❋
Luno	✓	✓	✓	✗	✓	❖	✙
Mycelium	✓	✓	✓	✗	✠	❖	❋
Unocoin	✓	✓	✓	✗	✓	❖	❋
Wirex	✓	✓	✓	✗	✓	❋	❋
Xapo	✓	✓	✓	✗	✓	❖	❋
Zebpay	✓	✓	✓	✗	✓	❖	❋

✓ Available ✗ Not available ❋ On app's home ♣ Under individual wallet/currency
✙ Under dedicated menu for wallets' summary ❖ Under dedicated menu for transaction history
✠ Redirects to an external website, leaving the app

Table 3.3: Actions available on Bitcoin wallet apps

From Table 3.3, it is clear that only three actions, i.e., open the app, receive Bitcoin, and send Bitcoin are available across all the wallet apps. Hence, we choose these three actions for classification, which indeed are the most important actions for Bitcoin transactions. It is also important to mention that opening the app may also be seen as the user's intent to inquire about the available balance or to synchronize transaction history. Given the wide-variety of distinct apps in the non-Bitcoin app category, we collected traffic traces for such apps while using each device normally for 8 hours. Next, we explain our equipment setup for experiments and data collection.

3.2.2 Equipment Setup

Figure 3.1 shows our equipment setup for collecting network traffic generated from the apps. The workstation was equipped with two Ethernet-based Network Interface Controllers (NICs); one for connecting it to the Internet and the other one for connecting it to the Wi-Fi Access Point (AP). The workstation was configured to forward traffic between Wi-Fi AP and the Internet. The smartphones were provided access to the Internet over a wireless connection via Wi-Fi AP. It is important to mention that only one smartphone was connected to the Wi-Fi AP at a time. To simulate user actions on the smartphones and thus evoke network traffic, scripted-commands were sent via USB. For the Android device, we used Android Debug Bridge (adb[1]), while for iOS device, we used Alloy 2.1.1[2] app that allows to automate the device without jailbreaking it. The generated network traffic was captured on the workstation using Wireshark 2.2.6[3].

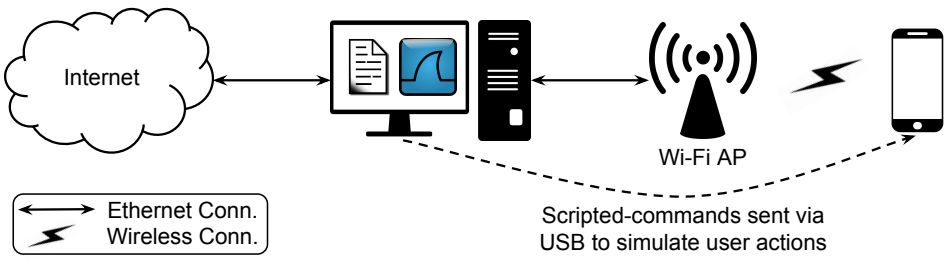


Figure 3.1: Equipment setup

The captured traces were exported to Comma Separated Value (CSV) files; each row holding the details of one captured packet. For each packet, we collected time, source IP address, destination IP address, ports, packet length, protocol, and TCP/IP flags. Although the packet's payload was gathered, it was discarded since it may or may not be encrypted. Finally, to

[1]https://tinyurl.com/y643gk3h
[2]https://tinyurl.com/yxrropge
[3]https://tinyurl.com/b86zcb

make the experiment scenario as close to the real environment as possible, we used only physical hardware and omitted any device emulator or virtual machine. We used Actiona 3.9.1[4] tool to coordinate data collection process.

3.3 Classifier Design

In this section, we present the design of our classification procedure. At first, we describe the data preprocessing phase necessary to get suitable training instances for the classification algorithm. Then, we briefly define the machine learning methods we used and how they have been trained and finally used for the classification.

3.3.1 Data Preprocessing

To handle network traffic traces via machine learning models, we need to perform a preprocessing step. In this work, we employ a procedure inspired by [201]. The complete procedure is composed of the following steps:

1. **Network trace capture:** The network trace capturing process aims to collect traffic data from a network in which simulated users are using apps connected to such network. The full equipment setup has been described previously in Section 3.2.2.

2. **Traffic *burstification*:** After the data collection phase, the network traffic is parsed. The parsing aims to obtain chunks of traffic that can be directly converted into training instances suitable for learning models. The first part of the parsing step is the so-called traffic *burstification*: the network traffic is divided into macro-chunks called bursts. A burst is defined as a sequence of traffic packets, where each packet is either received or transmitted within a threshold of time. In our experiments, such threshold has been fixed to one second.

3. **Flows separation:** The next part of the parsing step further divides the bursts into chuncks, called flows, corresponding to traffic between pairs of IP addresses/port. Anytime the port information was not available the corresponding packet was discarded. Similarly, flows with less than three packets has also been discarded.

3.3.2 Feature Selection

After the data preprocessing stage, we obtain a set of flows; each of them corresponding to a particular action of a specific app. The final step consists of converting the flows into training instances. It is important to notice that a single action of an application can produce more than one flow, and hence

[4]https://tinyurl.com/y3462q7l

it can produce more than one training instance. For each flow, the following feature selection procedure has been performed:

- Assuming the local IP address as the target endpoint of the flow, we convert each packet into a number corresponding to its length in bytes. If the packet has been sent by the target, such number is negated. At the end of this step, the flow is converted into a sequence (time series) of integer numbers. It is worth to note that the lengths of such sequences of numbers are not uniform and hence, in general, are not directly suitable as training instances. Machine learning models usually require fixed length input instances.

- The extracted sequence is finally converted into a training instance via a statistical feature extraction procedure, which simply compute some statistics over the time series. The used statistics are: length of the series, minimum, maximum, mean, median, mode, variance, skewness, kurtosis, and percentile at 25%, 50% and 75%. These statistics are collected for the entire sequence, for the incoming packets only, and for the outgoing packets only. Hence, the resulting training instance has a dimension of 36 (12×3). Finally, if we called $\mathbf{x} \in \mathbb{R}^{36}$ the vector containing the statistics, the complete training instance is described by the pair $(\mathbf{x}, y)$, where y is the target classification value.

It is worth to mention that while source and destination IP addresses have been used for flows separation, they were not leveraged in any way during the classification. The full procedure, including the data preprocessing steps, is depicted in Figure 3.2.

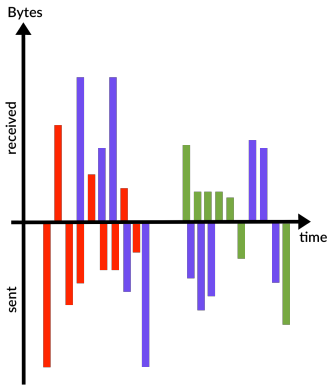
3.3.3 Machine Learning

Given the modularity of the proposed framework, any classification method can be plugged to perform the task. For experimental purposes, we employed two of the most successful machine learning methods for classification, namely Support Vector Machine (SVM) and Random Forest (RF).

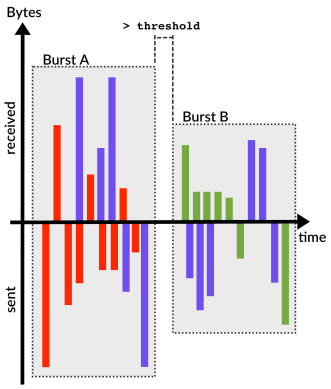
SVM is one of the most used kernel methods in machine learning [63, 185, 205]. Besides its good reputation in terms of classification accuracy, SVM's popularity is also given by its strong theoretical foundation. SVM aims to find a hyperplane that separates the instances with different labels. Such hyperplane is guaranteed to maximize the minimum distance between two points with different label. This property entail very good generalization capabilities to unseen examples, which is a desired feature for any machine learning model.

RF also known as random decision forests [116], are learning algorithm of the family of ensemble learning methods. RF for classification operate

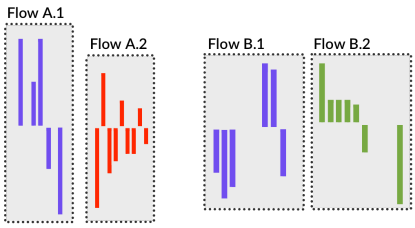
by constructing a set of decision trees at training time, and at the prediction time, they output the class that is the mode of the classes (classification) of the individual decision trees. One of the strengths of RF are their efficiency and their simplicity. Since they are based on decision trees, it is very easy to grasp what they do under the hood. Moreover, RF have achieved state-of-the-art performances in many classification tasks.



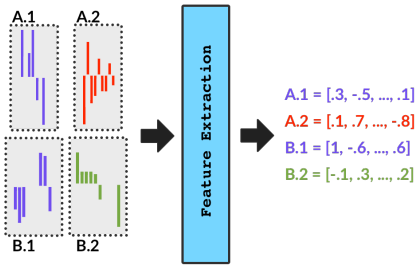
(a) Network traces capture: different colors represent different applications.



(b) Traffic *burstification*: traces are split into bursts.



(c) Flows separation: for each burst, different flows are separated by means of the pairs of source-destination IPs.



(d) Training set creation: flows are converted into vectors of statistical features about the packets length.

Figure 3.2: Network traffic preprocessing

3.3.4 Training

The training phase consists of learning the model parameters using the training set. In our scenario, in which we want to identify whether a Bitcoin app is being used and also which specific action of such app is being performed, we

need to tackle the problem at different levels. We can identify the following layers of classification:

1. Classify whether the instance represents a flow of a Bitcoin app or not;
2. If so, classify whether it belongs to an Android app or an iOS app;
3. If is categorized as Android (or iOS) app, classify the specific app;
4. Given the app from the previous step, classify the specific action.

The full stack of classification layers is depicted in Figure 3.3.

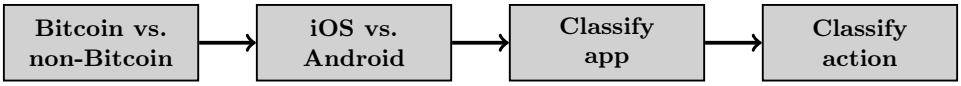


Figure 3.3: Classification hierarchy: (i) Bitcoin apps are isolated from the non-Bitcoin ones; (ii) Bitcoin apps are classified on the basis of the operating system; (iii) For the target operating system, the Bitcoin app is identified; (iv) given the app, the related action is identified.

Hence, the training phase requires to learn one model for each of the classification tasks described above. Before starting the training, we apply to the training instances a scaling function to normalize the input data. In particular, we used both MinMax and Standard scaling techniques. See Appendix B for further details on these scaling techniques.

3.3.5 Prediction

Given a new instance to classify, the prediction is performed using the same steps as in the training phase. Clearly, if a wrong prediction is made in one step, all the following steps will also be wrong except for user action classification, which can be still correct. This is due to the fact that same actions are shared between the applications, and it can be correctly identified even if the app identification is not correct.

3.4 Evaluation

In this section, we show the evaluation procedure used to assess the quality of the proposed approach. For each of the classification step identified in Section 3.3.4, we first trained a classifier, and then we tested it on an hold out set of instances. We performed two different experiments:

1. Single classifier assessment: in this setting, each single classifier is tested independently of the others.

2. Full stack classification: in this setting, the classification is performed following the full-sequence of classification as described in Figure 3.3.

Here, we also argue about the obtained results.

3.4.1 Evaluation Settings

All the experiments have been conducted using a stratified 5-fold cross validation. In order to increase the statistical significance of the result, we repeated each experiment 10 times with different 90-10% training and test splits. The validation procedure is used to do model selection, and the validated hyper-parameters for SVM and RF are shown in Table 3.4 and Table 3.5, respectively. We chose standard range of values for the hyper-parameters [117]. We also validated the scaling techniques shown previously.

Parameter	Validated values	Effect on the model
γ	$\{10^{-6}, \ldots, 10^{3}\}$	Shape parameter of the RBF kernel which defines how an example influence in the final classification.
C	$\{10^{-3}, \ldots, 10^{3}\}$	Regularization parameter that controls the trade-off between the achieving a low training error and a low testing error that is the ability to generalize your classifier to unseen data.

Table 3.4: Hyper-parameters validated for Support Vector Machine

Parameter	Validated values	Effect on the model
# of trees	$\{10, 50, 100\}$	Number of trees use in the ensemble.
Max depth	3, ∞	Maximum depth of the trees.
Bootstrap	yes / no	Bootstrap Aggregation (a.k.a. bagging) is a technique that reduces model variances (overfitting) and improves the outcome of learning on limited sample or unstable datasets.
Split criterion	gini, entropy	Criterion used to split a node in a decision tree.

Table 3.5: Hyper-parameters validated for Random Forest

It is worth to mention that, even though the dataset has been collected in a controlled setup, the full hierarchical classification well simulate a real-world scenario in which instances are gathered in real time. Table 3.6 describes the instances distribution over the apps and over the actions for both iOS and Android systems. The total number of instances for non-Bitcoin app are 4662. After the preprocessing step, Luno app did not produce any meaningful flow. Hence, we had to discard Luno app. One of the possible reasons for such behavior of the app could be that the app mostly processes the data off-line.

App	OS	Open App	Receive Bitcoin	Send Bitcoin	Total
BTC.com	Android	149	22	20	191
Bitcoin Wallet (Bitcoin.com)	Android	51	21	46	118
Coinbase	Android	286	20	19	325
Mycelium	Android	251	20	38	309
BitPay	iOS	20	54	20	94
Bitcoin Wallet (Bitcoin.com)	iOS	49	40	137	226
Blockchain	iOS	346	40	167	553
Bread	iOS	29	208	217	454
Copay	iOS	20	52	20	92
Total		1101	477	684	2362

Table 3.6: Dataset description: app name; operating system; number of instances for open app, receive Bitcoin, and send Bitcoin actions; and the total number of instances

It is worth to notice that the real proportion of Bitcoin and non-Bitcoin apps is actually much more imbalanced. However, in machine learning one of the standard approach to deal with highly imbalanced datasets is to use over-sampling of the minority class [53]. Thus, in our setting, the almost balanced dataset follows the same direction of the over-sampling technique. All methods have been evaluated using standard classification metrics: Accuracy, Precision, Recall, and F1 measure. See Appendix B for details on these metrics.

3.4.2 Results

In this section, we present and discuss the results obtained by our proposal on user activity identification task.

Single classifier assessment

The following battery of experiments aim to assess each classification layer individually. In these cases, every layer works with a controlled training set and independently from the others. The goal of this preliminary assessment is to check whether some of the classifications are harder than others. Table 3.7 to Table 3.12 present the results for single classifier for different tasks mentioned in Section 3.3.4. Here, we present the classification performances of RF and SVM over 10 runs of a stratified 5-fold cross validation. We report the average results with their standard deviations, and $(\cdot)$ indicates the best result for the metric.

For Bitcoin vs. non-Bitcoin app classification, we achieved an accuracy of 97.7% using RF, see Table 3.7. Next, as shown in Table 3.8, we attained an accuracy of 98.4% using RF in correctly identifying the OS to which an

app belongs to. The performance metrics for Bitcoin app classification on Android platform are listed in Table 3.9. Here, we reached an accuracy of 96.6% using RF. The accuracy in identification of user actions in Bitcoin apps on Android platform is listed in Table 3.10. The performance metrics for Bitcoin app classification on iOS platform are listed in Table 3.11. Here, we attained an accuracy of 96.2% using RF. The accuracy in identification of user actions in Bitcoin apps on iOS platform is listed in Table 3.12.

Method	Accuracy	Precision	Recall	F1
RF	0.977 ± 0.005·	0.977 ± 0.005·	0.973 ± 0.005·	0.975 ± 0.005·
SVM	0.930 ± 0.01	0.922 ± 0.01	0.923 ± 0.01	0.922 ± 0.02

Table 3.7: Bitcoin vs. non-Bitcoin app classification

Method	Accuracy	Precision	Recall	F1
RF	0.984 ± 0.01·	0.984 ± 0.01·	0.983 ± 0.01·	0.983 ± 0.01·
SVM	0.956 ± 0.01	0.955 ± 0.02	0.955 ± 0.02	0.955 ± 0.02

Table 3.8: App's OS classification

Method	Accuracy	Precision	Recall	F1
RF	0.966 ± 0.01·	0.968 ± 0.01·	0.968 ± 0.01·	0.968 ± 0.01·
SVM	0.945 ± 0.02	0.948 ± 0.02	0.948 ± 0.02	0.948 ± 0.02

Table 3.9: Bitcoin app classification on Android

Method	Bitcoin Wallet (Bitcoin.com)	BTC.com	Coinbase	Mycelium
RF	0.8 ± 0.15	0.98 ± 0.03·	0.991 ± 0.01·	0.971 ± 0.03·
SVM	0.85 ± 0.1·	0.975 ± 0.03	0.988 ± 0.02	0.958 ± 0.05

Table 3.10: Classification of user actions in Bitcoin apps on Android

Method	Accuracy	Precision	Recall	F1
RF	0.962 ± 0.02·	0.964 ± 0.02·	0.963 ± 0.02·	0.963 ± 0.02·
SVM	0.935 ± 0.02	0.938 ± 0.02	0.935 ± 0.02	0.935 ± 0.02

Table 3.11: Bitcoin app classification on iOS

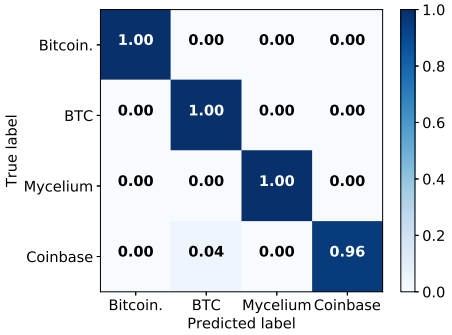
Method	Bitcoin Wallet (Bitcoin.com)	BitPay	Blockchain	Bread	Copay
RF	1.0 ± 0.0·	1.0 ± 0.0·	0.920 ± 0.02·	0.943 ± 0.03	1.0 ± 0.0·
SVM	1.0 ± 0.0·	1.0 ± 0.0·	0.911 ± 0.03	0.958 ± 0.04·	1.0 ± 0.0·

Table 3.12: Classification of user actions in Bitcoin apps on iOS

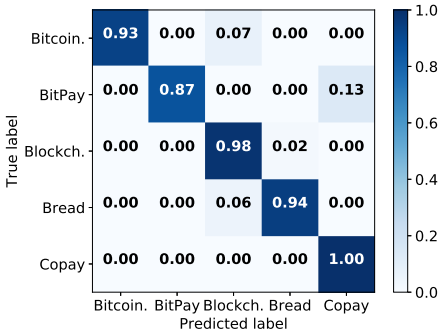
As discussed above, we obtained slightly better results on Android for Bitcoin app identification while user action identification was better on iOS. Moreover, RF performed better over SVM in most of the tasks.

Full stack classification

This experiment represents a simulation of a real-world scenario. Hence, the identification of both Bitcoin app and Bitcoin-related user operations have to be done assuming that the classifications made in the previous layers of the classification stack are correct. A single error in any stage of the classification stack influences all the subsequent ones. Figure 3.4 depicts the confusion matrix for identification of Bitcoin apps using RF for Android (Figure 3.4 (a)) and iOS (Figure 3.4 (b)). Similar to the previous experiment setting, we got better results on Android in Bitcoin app identification. Figure 3.5 (a) shows the confusion matrix for classification of user actions in Bitcoin apps using RF. Figure 3.5 (b) gives a comparison of accuracy achieved by RF and SVM along the full stack classification.



(a) Android

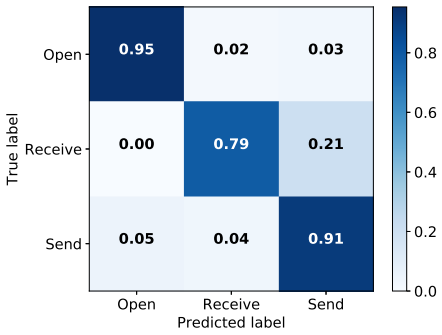


(b) iOS

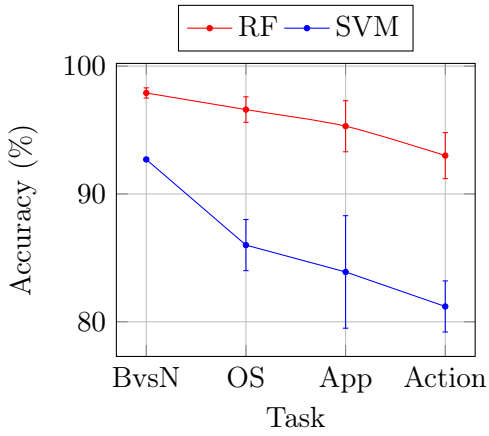
Figure 3.4: Confusion matrix for Bitcoin app classification using RF for both (a) Android and (b) iOS. The confusion matrices are taken from one out of the ten runs performed.

It is clear from Figure 3.5(b) that misclassification errors in one stage are propagates to the subsequent stages. Hence, the results for a stage of full stack classification are limited by the performance of the previous

stage(s); except for user action classification, which can be still correct (see Section 3.3.5). Nevertheless, we obtained an accuracy of nearly 95% in user action identification using RF in the full stack classification.



(a) Confusion Matrix



(b) Classification accuracy

Figure 3.5: (a) Confusion matrix for user action classification in Bitcoin apps using RF. The confusion matrices are taken from one out of ten runs performed. (b) Accuracy of both RF and SVM along the classification stack. Performance are reported as the average accuracy (%) over 10 runs.

3.5 Summary

The popularity of cryptocurrencies, especially Bitcoin, is increasing day-by-day. Bitcoin is now recognized as a regular mode of payment. The convenience of smartphones has also driven people to adopt and use this new currency. In this chapter, we have focused on identification of user actions within Bitcoin wallet apps. By analyzing network traffic using machine learning techniques, we have identified the most crucial user actions related to Bitcoin transactions with a very high accuracy of nearly 95%.

Part II

Algorand

Chapter 4

Security Flaw in Truly Democratic Consensus Protocol

The blockchain technology has been adopted by almost every scenario in today's digital world with an aim to keep temper-proof records [59]. Originally, the concept of distributed consensus - upon which the blockchain technology is founded - dates back to the '90s [83, 103, 112, 199]. But, after the success of Bitcoin, a multitude of other areas have been transformed by the idea of a transparent and distributed public ledger, such as elections, contract management, insurance, and charity [16].

The Bitcoin system combines together different well-known concepts including digital signatures, hashing functions, Proof-of-Work (PoW), and Merkle trees. However, with an exponential growth in the number of users and Bitcoin-miners, the Bitcoin network has observed collateral effects of its consensus protocol. In particular, it suffers from scalability [66], hashing power centralization, mining strategies exposures [100, 142, 179], and energy- and computational-deficiency [162]. To this end, a variety of solutions have been proposed over time. Some of these solutions aim to improve the existing version of PoW (e.g., Litecoin[1], Primecoin[2] [132]) while other proposals used completely different concepts, such as Proof-of-Stake (PoS) [8, 133], Proof-of-Burn (PoB) [163, 191], and Proof-of-Elapsed-Time (PoET) [10]. As a representative example, PoS systems (both stake modification-based [7, 133, 166, 176] and actual balance-based [8, 206]) have critical concerns from the security point of view [137]; such issues have been partially solved by providing formally-proved security guarantees [40, 131] or by discouraging users to act maliciously through punitive procedures [48, 49]. Other works [167, 187]

[1]Litecoin adopted its *scrypt* PoW function's design to increase the decentralization of the system and further optimize the mining process [165].

[2]Primecoin exploits the computational power used for the mining activity to contribute to mathematical research works.

tried to achieve better scalability and throughput via a block graph structure based on Directed Acyclic Graph (DAG), albeit, these solutions have their own demerits.

On the other side, Algorand [54, 106] - an innovative protocol for the blockchain technology - has been proposed to overcome these limitations. Algorand not only guarantees an overwhelming probability of linearity of the blockchain, but it also aims to solve the "blockchain trilemma" of decentralization, scalability, and security. Algorand uses a process, called *cryptographic sortition*, to securely and unpredictably elect a set of voters from the network periodically. These voters are responsible for reaching consensus through a Byzantine Agreement (BA) protocol, for one block at a time. It guarantees an overwhelming probability of the blockchain's linearity and a block generation time of nearly a minute. It is comparable to the approaches presented in the works [40, 131] and has the potential to shape the future of the blockchain technology.

Given the promising properties of the Algorand protocol in terms of decentralization and scalability, the security aspects of its consensus algorithm are crucial. To the best of our knowledge, our work is the first analysis of Algorand's security. We aim to at least start a discussion about a possible security flaws in the message validation process, which can possibly expose honest nodes' bandwidth and memory resources to Distributed Denial-of-Service (DDoS) attacks.

Contributions: The major contributions of our work are as follows:

1. We demonstrate a practically feasible attack on the Algorand protocol. The attack has the potential to target an arbitrarily sized group of honest users and slow down the consensus process; leaving the targeted nodes behind in the chain as compared to the non-attacked nodes.

2. We implemented the protocol in our Java-based simulator and evaluated the feasibility of our attack. The simulator is available on request.

Organization: The rest of the chapter is organized as follows: Section 4.1 gives a thorough description of the Algorand protocol. We elucidate our attack scenario in Section 4.2. Section 4.3 presents the implementation details of our simulator, evaluation settings, and our results. Section 4.4 discusses the feasibility of our attack and the possible countermeasures. Finally, Section 4.5 summarizes the chapter.

4.1 Algorand

Since the main focus of our work is on Algorand, we being by introducing the Algorand protocol. We elucidate the main limitations of current blockchain

technologies in Section 4.1.1 and present the distinct features of Algorand in Section 4.1.2. Section 4.1.3 discusses the assumptions specified by the Algorand protocol with respect to the adversary model. Section 4.1.4 gives an overview of the network topology, communication protocol, and description of messages used by Algorand. Then, consensus mechanism of Algorand is explained in Section 4.1.5. The technical details of cryptographic sortition, a critical component of the consensus process, are discussed in Section 4.1.6.

4.1.1 The Limitations of Current Blockchain

The main motivation for the development of Algorand lies in the underlying assumption and technical problems that are essentially shared by most of the PoW-based blockchains [54]. Those assumption and limitations include:

1. [Assumption] Majority of nodes contributing to the computational power of the network will be honest.

2. [Technical Problem] Wastage of computational power as well as electrical energy.

3. [Technical Problem] Concentration of power, i.e., the total computing power for block generation lies within just few mining pools.

4. [Technical Problem] Ambiguity in the consensus reached by different network nodes on the confirmed transactions, which leads to a fork in the blockchain.

4.1.2 Salient Features of Algorand

The protocol has the following distinct and promising features:

1. It is designed to be fully decentralized and democratic. Moreover, there is no distinction between the role of different groups of users, e.g., miners vs. "normal" users.

2. Each user runs the same functions with negligible hardware requirements (i.e., with negligible computational effort) as the concept of miner/mining does not exist in Algorand.

3. It is scalable with the number of nodes and with the confirmed transactions throughput. The authors [106] have shown that the throughput of Algorand is nearly fifty times that of the Bitcoin system.

4. The probability of forking is practically zero ($P_{fork} = 10^{-12}$).

4.1.3 Protocol Assumptions

To guarantee security of the blockchain, Algorand relies on the conventional assumption of PoS systems, i.e., Honest Majority of Money (HMM). In particular, Algorand assumes a continuum in the ownership of currency belonging to honest users, i.e., at any round r, at least a fraction h (greater than two-third) of the money held by honest accounts on round $r - k$ must still belong to the honest users, for some integers h and k that are defined by the protocol. Together with HMM, Algorand requires a partial synchronization assumption on the honest nodes' clocks. It states that the honest nodes' clocks are not required to be strictly synchronized, but they are assumed to have the same speed with a bounded tolerance.

Furthermore, Algorand was built to face a very strong adversary. The adversary is capable to perfectly synchronize all malicious nodes that he[3] controls and can corrupt honest users indiscriminately and instantly. However, the attacker's capabilities are bounded by the following three main facts: (1) he can corrupt honest users until the HMM property is maintained; (2) he cannot forge secret keys of the honest users that he has not corrupted; and (3) he cannot prevent an honest recipient to receive messages.

4.1.4 Network Communication

The Algorand network is a Bitcoin-like peer-to-peer network. Each user is identified by a public/private key pair, and each key pair corresponds to a node in the topology. For simplicity, we will use the terms "user" and "node" interchangeably. Each node establishes and maintains a different TCP connection with each of his peers. The current specifications of Algorand do not specify if a node has to choose his peers based on their stake, which indeed exposes the network to a Sybil attack[4].

Nodes broadcast messages through the network using a standard gossip protocol - each node gossips his message to his peers, who in turn relays it to their neighbors. A message has to be validated before it can be relayed, and it is never sent twice to the same user. The protocol defines four types of messages: (1) transactions; (2) voting messages; (3) block proposals; and (4) credential messages.

A transaction (t) from user i to user j is defined as:

$$t = sig_i(pk_i, pk_j, a, I, H(SI)), \tag{4.1}$$

where pk_u is the public key of the user u, a stands for the amount of Algorand units to transfer from pk_i to pk_j, I is an optional string to describe the transfer, $H(SI)$ is a 256-bit hash produced by hashing a secret string SI (not

[3]We refer to different entities as masculine entity without any gender-bias.

[4]In a sybil attack, the attacker creates a large number of nodes at zero cost (i.e., with no stake) to increase the probability of connecting with his targets.

specified in the body of the message), and $sig_u(x)$ denotes digitally signed data x using the private key of user u.

A voting message (vm) from a user i in round r and step s is defined as:

$$vm_i^{r,s} = (sig(v_i^{r,s}), \sigma_i^{r,s}), \tag{4.2}$$

where $\sigma_i^{r,s}$ represents the sortition proof of the user i in s^{th} step of r^{th} round (further details in Section 4.1.6), $v_i^{r,s}$ defines the vote of user i in round r and step s.

A block proposal (bp) for round r from user i is defined as:

$$bp_i^r = (B_i^r, sig_i(H(B_i^r)), \sigma_i^{r,1}), \tag{4.3}$$

where B_i^r is defined as:

$$B_i^r = \begin{cases} (r, PAY^r, sig_i(Q^{r-1}), H(B^{r-1})), & \text{if } B_i^r \neq E^r; \\ (r, Q^{r-1}, H(B^{r-1})), & \text{otherwise;} \end{cases} \tag{4.4}$$

where PAY^r is a set of transactions t, and Q^r is called *seed* and defined as:

$$Q^r = \begin{cases} H(sig_{l^r}(Q^{r-1}, r)), & \text{if } B^r \neq E^r; \\ H(Q^{r-1}, r), & \text{otherwise;} \end{cases} \tag{4.5}$$

where l^r is the user who created the consensus block B^r for round r. The seed is a parameter needed in the sortition process (more details in Section 4.1.6). E^r is the default block for round r, and it is defined as:

$$E^r = (r, Q^{r-1}, H(B^{r-1})). \tag{4.6}$$

Finally, credential message (cm) from a user i for a round r is defined as:

$$cm_i^r = (\sigma_i^{r,1}). \tag{4.7}$$

It consists of the sortition proof of the block proposer (in this case, user i). Each sortition proof is associated with a priority value in a deterministic way (further details in Section 4.1.6). Credential messages are far smaller in size than the block proposals. Thus, they propagate faster through the network. Credential messages are gossiped by the block proposers at the beginning of a round along with their block proposals. Since the block proposals and the credential messages from the same block proposer contain the same sortition proof, the peer nodes can leverage the priority values from the credential messages to not relay block proposals with low priorities. This helps in preventing congestion in the network.

4.1.5 Consensus Algorithm

Algorand's consensus algorithm consists of a synchronous protocol that combines the concepts of PoS systems with a Byzantine fault tolerance agreement. The protocol virtually schedules the time into rounds. At each round, all the network nodes attempt to reach consensus on a new block of transactions. Each round is composed of the following actions:

- Each node in the network must first check its role to determine whether he has to propose a block for a given round. To do so, the nodes use cryptographic sortition, which requires them to run a trivial (a single hash) computational challenge. In case a node has to propose a block, he collects all the pending transactions inside a block proposal and gossips it together with the sortition proof. The block's size is limited by a fixed parameter in the protocol.
- Each node waits for incoming block proposals from the previous step for a predefined duration of time. Among all the valid blocks he receives, he selects the one with the highest priority sortition proof. Then, he computes an hash using this block and sets this hash as the input for the BA of the next steps.
- All nodes in the network try to reach consensus on one block through a BA protocol. The BA has two key phases. Both the phases are composed by subsequent steps. At each step, a distinct group of users (called committee members) is elected in an unpredictable way through cryptographic sortition. These committee members spread their voting messages based on votes received from the previous step. The committee members attach a sortition proof that proves their legitimacy while spreading their voting messages.

The two phases of BA are as follows:

1. The first phase is called the *reduction* phase that comprises two steps. In the first step, each committee member votes for the hash of the blocks proposed for consideration. Next step, committee members vote for the hash that received votes over a certain threshold (defined by protocol). In case none of the hash/block receives enough votes, committee members vote for the hash of the default empty block.
2. The next phase does another binary BA to ensure that each node agrees on the same consensus; the consensus can conclude on the output of the previous phase or on the default value in the absence of a majority.

4.1.6 Cryptographic Sortition

Cryptographic sortition is used by each network node at the beginning of every step of each phase to determine whether he has a role in that particular

step. It is a simple and lightweight process that involves computing a single hash and a digital signature. The sortition algorithm is implemented using a Verifiable Random Function (VRF[5]) [150]. In particular, the hash for a certain step s and round r is computed by the user i as:

$$y = H(sig_i(r, s, Q^{r-1})), \tag{4.8}$$

where H is a hashing function, y a 256-bit long hash, and Q^{r-1} is the seed quantity defined by Eq. 4.5. The quantity $sig_i(r, s, Q^{r-1})$ is called the sortition proof and is attached to voting messages or block proposals to prove the legitimacy of the corresponding voters or block proposers. The hash is used by the nodes to verify sortition condition. The possibility that the condition is verified is directly proportional to the stake of the node and a constant parameter π_{role} that is fixed in the protocol, which guarantees that on average a fixed number of nodes are elected at each step for a certain role. For this reason, the protocol allows users to be elected more than once in the same step. A user i can possibly be elected at each step w_i times, where w_i is the number of user's units of currency. Formally, the interval $[0, 1)$ is partitioned into w_i intervals I, where the j-th interval I_j is defined as:

$$I_j = \Big[\sum_{x=0}^{j} B(x; w_i; p), \sum_{x=0}^{j+1} B(x; w_i; p)\Big); j \in (0, 1, ...w_i), \tag{4.9}$$

where $B(x; w_i; p)$ is the binomial probability that user i is elected exactly x times on w_i chances, each with probability $p = \pi_{role}/W$. Here, π_{role} estimates the expected number of sorted users for that role and W is the total amount of currency in the network. The number of votes are found by applying the binary point operator to the hash (i.e., $y/2^{y.length}$) and searching the interval I_j with which the value is associated. If the value lies on the j-th interval, then i can vote j times. There are at least three interesting observations about this procedure:

1. Since on average a fixed number of nodes are elected at each step, the traffic of voting messages in the network does not increase with the number of participants.

2. The procedure to find if a user i is sorted for a certain step in a round can be executed only by the user i as it involves a signing process.

3. Since Q^{r-1} can be computed by a node only when that node reaches consensus on round $r-1$, the result of the sortition of user i in a certain step of round r is unpredictable and unknown even to the user i. This avoids the possibility of collusion between voters of one or more steps to manipulate the consensus process on a certain round.

[5]VRF functions allow users to produce verifiable computations. A user produces a proof using his private keys that can be verified by other users via the producer's public key.

4.2 Our Attack

We explain our attack in this section, starting from the attack preliminaries in Section 4.2.1 followed by a brief introduction to the flooding attacks in Section 4.2.2 and the description of our attack in Section 4.2.3.

4.2.1 Attack Preliminaries

In our adversary model, we assume that all the malicious nodes are coordinated by a single/unique entity. This translates into the following two properties of the malicious nodes:

1. Each malicious node can sign messages using any other malicious node's private key.

2. The malicious nodes participate in the protocol either passively or actively. These malicious nodes coordinate and use the messages they receive from their honest peers to become aware of the protocol's execution status.

We assume that all the malicious nodes and their public/private key pairs were created by the adversary sufficiently in advance, i.e., before the attack takes place. This enables all of them to become the voters or block proposers in the protocol[6]. As mentioned in Section 4.1.4, we can also safely assume that the network is vulnerable to the Sybil attack, meaning that the honest nodes choose their peers randomly without relying on their stake. In this way, the malicious nodes have the same probability as of the other nodes to connect to honest peers. On the other side, honest peers can control/limit the number of incoming connections, denoted by *max-connections*.

4.2.2 A Typical Flooding Attack

In a typical network scenario, a flooding attack targets honest nodes' bandwidth and memory resources by sending a huge number of message to him. Typically, a flooding attack sends a huge number of fake or invalid messages towards the target nodes to slow down their message reception and message processing. Crafting such attacks from a single nodes is not very effective because nodes in Bitcoin-like decentralized networks - where no level of trust is assumed between participants - often rely on "misbehaviour" scores to label and identify possible malicious users. As soon as an honest node finds that one of his peers is malicious, he can stop processing messages from that peer, drop the connection, and blacklist him to prevent future connection

[6]This assumption is necessary because Algorand increases the unpredictability of the outcome of the sortition process by limiting the set of keys that can participate in the process. The eligible keys must be created at least k rounds before.

requests from that peer for a certain time. Hence, the attacker needs a large number of malicious nodes connected to the target. So, he can send some messages from each connection in a coordinated manner and create the same impact without being detected (or at least fully detected).

4.2.3 Magnifying Attack's Impact via Undecidable Messages

Our attack aims to increase the impact of the flooding attacks by exploiting the message validation process of Algorand. Due its design, our attack also enables an adversary to have a longer connection time before getting blacklisted by the target node. In Algorand, each message that is a part of the consensus algorithm (i.e., voting messages (Eq. 4.2) and block proposals (Eq. 4.3)) is subject to two distinct checks during the validation process:

1. *Stateless checks*: A set of checks on a message/packet that a validator node can perform to know its current status (e.g., packet's structure, packet's authentication, check for duplicated messages).

2. *Sortition proof check*: The sortition proof must be checked to verify that the sender/author of the message had the right to gossip it.

As described Section 4.1.6, the sortition proof is a byte-string that represents the signature of a user i in the form $sig_i(r, s, Q^{r-1})$, the validator node should hash this string to verify whether the given output satisfies the sortition property. However, before blindly hashing the signature, the protocol requires him to verify it first. Thus, he needs all the parameters - that were originally used to create it - including the seed quantity Q^{r-1}. This is a stateful check because the validator should have already reached consensus on the round $r-1$ to independently compute Q^{r-1}. If that is not the case, then the message cannot be fully validated at the current status of the validator node. In such a situation, the only option that the validator node has is to store the message and wait until it can be fully validated because messages that are not yet fully validated, cannot be discarded. We refer to these messages as *undecidable messages*.

At this point, the aim of the attacker is to flood honest targets with as many undecidable messages as possible to saturate their bandwidth and possibly their memory. These two objectives decide the type and number of messages used to construct our attack vector. With respect to the type, we chose the largest possible message, i.e., block proposals containing max-sized blocks. Since messages need to pass the stateless checks, each public key can gossip only a single block proposal per round. However, keys are created at zero cost and each malicious node can sign messages with any of

the adversary keys. Hence, each node can include an arbitrary number of block proposals in his payload - one for each key[7].

Now, we describe the attack's execution in practice. The attacker connects to his target as one of his peers. Since the target node is assumed to honestly execute the Algorand protocol, as soon as he receives and validates a message for a certain round r, he has to relay it to all of his peers including the malicious node. Hence, the attacker just needs to (1) prepare its payload in advance[8] for the attack in round r; (2) wait to receive a block proposal or a credential message (Eq. 4.7) from his target[9] in round r to be aware of the fact that the target has started the execution of round r; and (3) send the payload to the target from each connected malicious node.

4.3 Evaluation

Here, we present the evaluation of our proposed attack. Section 4.3.1 provides the technical details of our simulator while our evaluation settings and results are presented in Section 4.3.2 and Section 4.3.3, respectively.

4.3.1 Simulator

Since, the source code or an official simulator for Algorand was not available at the time of our experiments, we built our own simulator that is completely written in Java. Both the works [54, 106] on Algorand propose slightly different, but overlapping versions of the protocol. The differences are in the terminating conditions to reach the final majority consensus within BA. However, our threat model is independent of the particulars of the chosen version of BA. We referred to the protocol described in the technical paper [106] for the implementation in our simulator.

[7] As long as the messages are authenticated and sent only once, a single public key can create a different undecidable message for every step in each round. It means that it can create a block proposal on the first step and a voting message for every step of BA (until the maximum m-th step allowed by the protocol). However, sending one message for every step implies that the given public key is sorted at every step of that round, which is very unlikely. A similar argument can be used by the target node to statistically label malicious users without waiting for their messages to be fully verifiable. Moreover, given a reasonable max-block size (order of MB) and m set to 150 steps [106], a single block proposal containing a max-size block would make up the large majority of the total payload weight. Hence, choosing a single block proposal for each key would imply that each user is elected only once in a given round, and cheaters are more difficult to identify a-priori.

[8] The attacker does not need any information to pre-compute a payload for a certain round r. He just needs to sign a random messages declaring them as if they belong to round $r + 1$. All other information will be checked only after the message becomes fully verifiable in round $r + 1$; by that time the attacker would have already achieved his goal.

[9] Block proposals together with the credential messages are gossiped first through the network at each round.

We model an honest node as an entity composed of three main components: (1) a network interface to send/receive messages from peers; (2) a validator process to verify the received messages; and (3) the process that runs the protocol. Since our attack focuses on flooding block proposals or voting messages, transactions are never sent/received directly in our simulation. Instead, the throughput (transactions confirmed per second) is simulated by creating block proposals containing a fixed number of arbitrary transactions[10].

To reduce the impact of simultaneous/parallel signature verification during simulation, we simulate block validation by using a sleep procedure. Furthermore, nodes are provided with a cache memory not to verify the same transaction twice[11]. While block signatures and other checks, e.g., packet authentication and sortition proof validation, are actually computed legitimately. We do not assume any latency in the network communication for the sake of presenting the minimum impact of our proposed attack, which would further enhance in the presence of latency.

4.3.2 Evaluation Settings

All simulations were done on a virtual machine on OpenStack [24] running Ubuntu 16.04 with 16 Intel Core Processors (Hashwell, no TSX, IBRS) and 64 GB of RAM. In our experiments, we simulated a network of 500 nodes, where each node had a 30 Mbps upload and 30 Mbps download bandwidth. The target nodes were connected to all the malicious nodes involved in the attack. However, the number of malicious nodes does not represent the *max-connections* parameter controlled by the honest nodes as honest nodes were connected to malicious as well as other honest peers. For the sake of simplicity, we assumed that once an undecidable message fails in the verification process, then all messages from the sender of that message are discarded without being processed. Each simulation was run for nearly 45 minutes, and the attack payload was prepared using different blocks sizes during different experiments.

To evaluate the effectiveness of our attack, we focus on: (1) the number of "legitimate" messages received and validated in due time by the honest nodes and (2) the average time taken to complete a round. These two metrics are co-related in a way that if sufficient messages are not received/processed in due time for a given step, the execution time of the corresponding step increases until a timeout is reached. We thoroughly evaluated our attack scenario by varying: (1) the number of malicious nodes involved in the attack;

[10]We assume that in a real implementation of the protocol, messages and transactions are processed by independent processes.

[11]Without a dedicated cache, transactions are validated at least twice. The time when the transaction is gossiped by its original creator and the time when it is received in a block.

(2) the number of block proposals sent from each malicious node, i.e., keys per malicious node; and (3) the size (in MB) of the attack payload.

To show the impact of our attack, we considered realistic settings where only a very low number of nodes (1% to 3%) are malicious in the network. The maximum number of block proposals sent from one malicious node at the time of attack was 70, following the analysis of the cryptographic sortition for block proposers by the authors of the protocol [106]. We set the block proposal receiving time and the step timeout (explained in the next section) to 150 seconds and 60 seconds, respectively.

4.3.3 Results

The execution time for a round consists of two parts: (1) block proposal receiving time and (2) step timeout. The block proposal receiving time is constant and is defined as the maximum window of time for the nodes to receive the block proposals for a given round. The step timeout is the maximum amount of time by which each step must complete. Here, each step finishes as soon as enough voting messages are processed or the step timeout occurs. In "no attack" scenario, the largest contribution to round time should come from the block proposal receiving time and the steps should execute before the step timeout occurs. The same is reflected in our results shown in Figure 4.1.

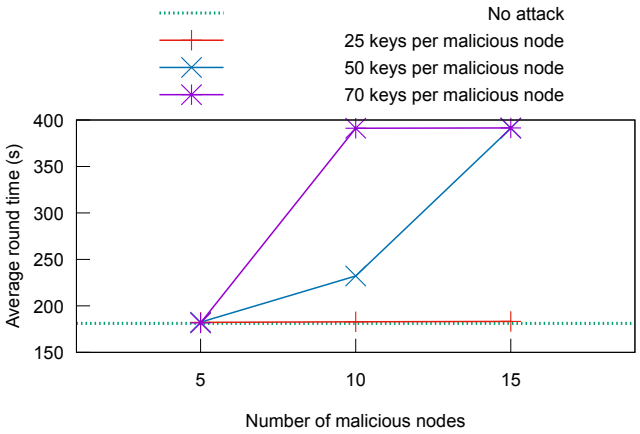


Figure 4.1: Effect of the number of malicious nodes and keys per malicious nodes upon average round time

Here, in the absence of malicious nodes, nearly 83% of the round time was taken by the fixed block proposal receiving window. On the other side, the round time remains nearly the same as "no attack" scenario with a lower number of malicious nodes/keys per malicious node. However, the protocol's performance starts to deteriorate with increasing number of malicious nodes/keys per malicious node, starting from the attack payload of just

500 blocks (10 malicious nodes with 50 malicious keys). In facts, the average round time with all the remaining higher attack configurations was over 390 seconds; meaning that these configurations caused at least four step timeouts in addition to the block proposal receiving time. As discussed earlier, our proposed attack does not prevent nodes from reaching consensus. Instead, it aims to significantly increase the execution time of each round, which leads the attacked nodes to reach their consensus on the default value, leaving the targeted nodes behind the non-attacked nodes in the chain.

The next important criteria to evaluate the impact of an attack on Algorand is the percentage of "legitimate" messages received and "legitimate" messages validated in due time. Figure 4.2 shows the effect of our attack upon "legitimate" messages received and validated with respect to the number of malicious nodes and keys per malicious nodes. Also here, with an attack payload of just 500 blocks (10 malicious nodes with 50 malicious keys), the percentage of "legitimate" messages received/validated starts to degrade considerably, which eventually lead to step failures.

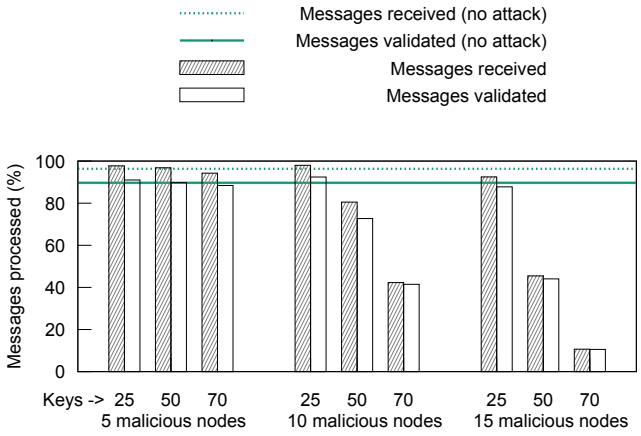
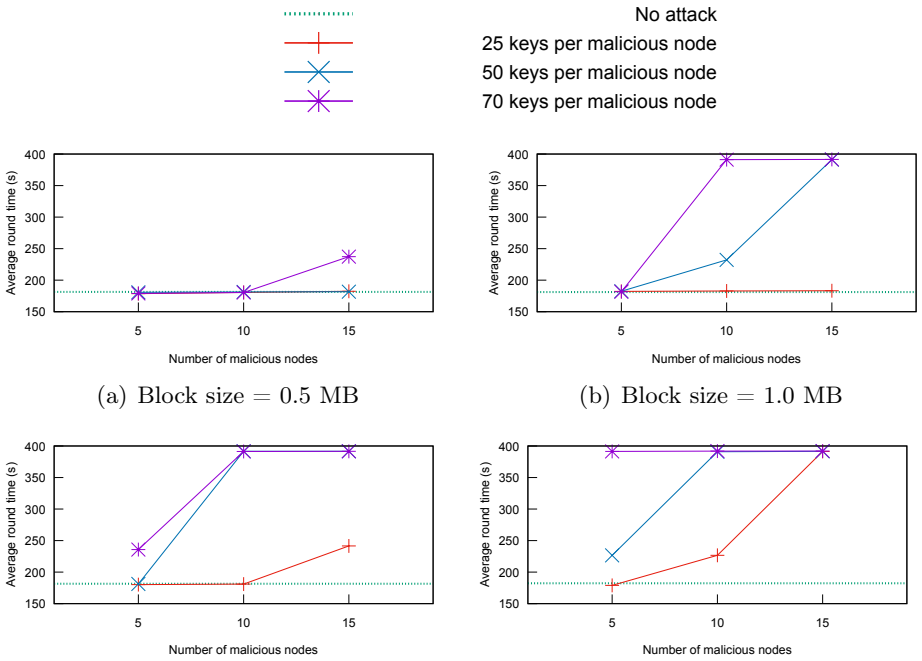


Figure 4.2: Effect of the number of malicious nodes and keys per malicious nodes upon percentage of "legitimate" messages received and validated

It is important to mention that an adversary can tune our attack on the number of malicious nodes/keys to prevent the detection while still creating significant disturbance in the network. Moreover, the size of the blocks is another important parameter to tune the attack. The results presented in the Figure 4.1 and Figure 4.2 were obtained using 1 MB of attack payload. Figure 4.3 and Figure 4.4 show the impact of attack payload's size upon the average round time and the percentage of "legitimate" messages received as well as validated in due time, respectively. Our results show that the size of the block does not significantly affect the performance of honest nodes in the absence of the attack. However, increasing size of the attack payload severely affect the performance of honest nodes in term of both the average round time and messages processed. It happens as the block's size directly affects

the download time of a block, which is further worsened by the number of malicious nodes and the keys per malicious nodes involved in the attack.



(a) Block size = 0.5 MB

(b) Block size = 1.0 MB

(c) Block size = 1.5 MB

(d) Block size = 2.0 MB

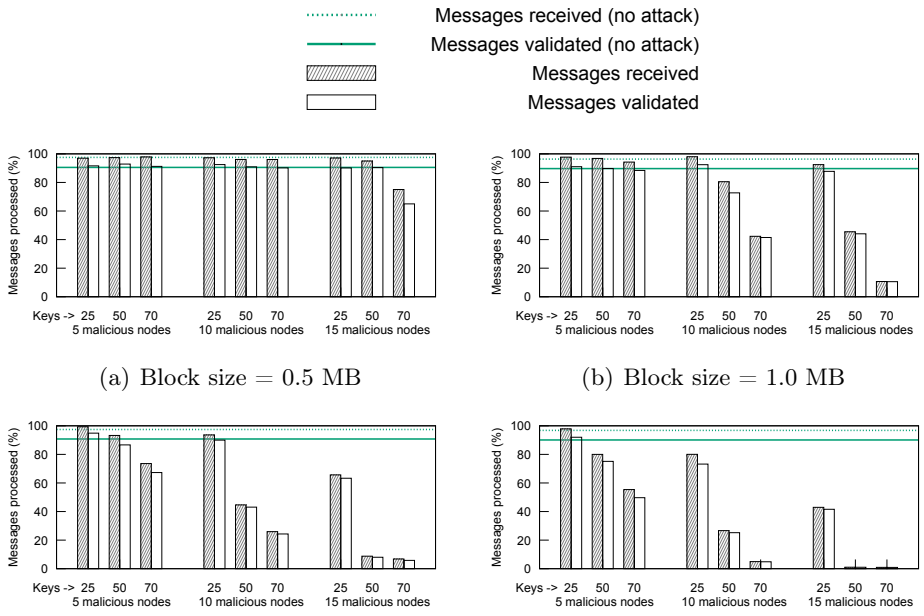
Figure 4.3: Effect of the attack payload's size upon average round time

4.4 Feasibility of the Attack

The attack costs practically nothing to an adversary. However, the adversary has to overcome one minor challenge, i.e., to connect at least one malicious node with the honest target. This malicious node can then flood the target with an arbitrarily large number of block proposals - one for each public key controlled by the adversary. However, the recipient can monitor and label connections from which he receives a suspiciously large number of messages for a round. Then, it is likely that the honest nodes would simply drop (or, at least temporarily stop listening) from such a connection. Hence, the success of our attack depends on how many malicious connections an attacker is able to establish with the target node.

The other important aspects of our attack are: (1) since TCP connections are required to be established, IP spoofing is not an option[12] and real

[12]Establishing TCP connections via IP spoofing is problematic as one cannot complete the TCP three-way handshake protocol. In particular, SYN-ACK packets from the target nodes are not sent to the real location of the malicious node. To make this happen, the adversary must have control over at least one router on the path between the fake address and the target, or the target node's network has allowed source-routing [194] of the messages, or the Initial Sequence Number (ISN) in the handshake protocol is vulnerable to the prediction attack. All of these speculations are quite strong assumptions.



(a) Block size = 0.5 MB

(b) Block size = 1.0 MB

(c) Block size = 1.5 MB

(d) Block size = 2.0 MB

Figure 4.4: Effect of the attack payload's size upon the percentage of "legitimate" messages received and validated

addresses should be used; (2) the target's *max-connections* parameter defines the maximum number of connections allowed towards it. We believe that the former issue can be solved by using botnet or relying on hidden services, such as Tor, to protect address identity inside the network. However, the *max-connections* parameter is out of the adversary's control. Hence, it is considered as an attack variable in our attack scenario. Consequently, establishing and maintaining a high number of connections with the target nodes is somehow a bit more challenging. But, the goal is achievable in the current design of the protocol, where peers are not weighted on their stake.

4.5 Summary

The Algorand protocol was proposed to overcome the limitations of conventional blockchain technologies. The protocol has great potential and is in under active development. In this chapter, we present a particular security flaw of Algorand protocol that exploits its process of validating messages. In particular, we evaluated a possible DDoS-like flooding scenario under practical assumptions, where honest nodes suffer significant delays in the execution of protocol. Furthermore, we also discuss the major factors that make this attack scenario more challenging for the honest nodes. Finally, we also present possible solutions to prevent such an attack.

Part III

Cryptominers

Chapter 5

Detecting Covert Miners via Magnetic Side-channel

Cryptomining (or, simply mining) is a process of validating and adding new transactions in the blockchain digital ledger for a given cryptocurrency. It is an essential process to keep most of the cryptocurrencies running. Typically, mining is a resource-intensive process that continuously performs heavy computations. Upon successful mining, cryptominers (or, simply miners) receive newly generated cryptocoins as their remuneration. Usually, newer cryptocurrencies tend to pay a higher reward. Some cryptocurrencies, such as Monero, make mining feasible on the web-browsers that enable even naive users to participate in the mining process.

After the success of Bitcoin, several alternative cryptocurrencies (Altcoins) have been introduced to the market. At the time of writing, there are over 2000 active cryptocurrencies [2]. The massive number of cryptocurrencies raises an enormous demand for mining. This demand continues to remain huge because mining, as mentioned before, is an inevitable operation to keep these virtual currency systems running. Such an immense demand for mining has also attracted cybercriminals [11, 28] to earn financial gains, who have already been exploiting cryptocurrencies to perform several types of financial crimes, e.g., ransomware [59].

Apart from investment in hardware, the cost of electricity to power up mining hardware and cooling facilities is one of the significant expenses associated with mining [22]. Mining popular cryptocurrencies, such as Bitcoin, is not profitable using personal resources (mainly electricity) unless the mining is done using specialized hardware [25]. However, mining can be lucrative if it is done using "plundered" resources, e.g., exploiting infrastructure at workplace. We can broadly categorize such plundering into two classes: (1) **conscious-mining** and (2) **unconscious-mining.** Conscious-miners exploit infrastructure allocated to them, e.g., an unethical employee

who mines at the workplace. Unconscious-miners mine unknowingly for a third party, e.g., a visitors to a website that hosts cryptomining scripts. Such exploitation of the computational resources causes financial damage - primarily in the form of increased[1] electricity bills - to the victims, who often discover the misuse when the damage has already been done.

On another side, a recent study [154] has suggested that "Bitcoin usage could alone produce enough CO_2 emissions to push global warming above 2 °C within less than three decades." The current situation would further worsen with illegal/unauthorized/covert cryptomining. Besides, unethical mining has lured all members of the modern society: government employees [13], corporate employees [26], students [18], teachers [30], researchers [9], nuclear scientists [23], and undoubtedly, hackers [11, 28].

To summarize, we can say that covert cryptomining has financial consequences, such as monetary losses, as well as societal influences such as ethical and psychological impacts. Moreover, a solution to detect covert cryptomining that focuses on a particular cryptocurrency may not be adequate to tackle the current situation, where new currencies emerge every day that gives miners a wide variety of options to choose from.

Contributions: In this chapter, we make the following contributions:

1. We propose a novel approach that leverages the magnetic side-channel to detect covert cryptomining. The scope of our approach is broader as we target the core of the mining process, i.e., the mining algorithms.

2. We implemented our approach and built a complete system. We included twelve different cryptocurrencies in our experiments, which indeed are the most mined cryptocurrencies.

3. We thoroughly evaluate the quality of our proposed approach. To this end, we designed and performed five different experiments: (1) *Binary* classification; (2) *Currency* classification; (3) *Nested* classification; (4) *Unseen-miner program* classification; and (5) *Cross-platform* classification. Our results show that our approach can reliably classify cryptomining activities.

Organization: The remainder of this chapter is organized as follows: Section 5.1 presents the essential concepts related to our work. Section 5.2 provides a comparative summary of the related works. We elucidate our system's architecture in Section 5.3 and its evaluation in Section 5.4. Section 5.5 addresses the potential limitations of our proposed approach. Finally, Section 5.6 summarizes the chapter.

[1] A machine consistently performs heavy computations while it does cryptomining, which, in turn, continuously draws electricity.

5.1 Background & Preliminaries

In this section, we briefly explain the fundamental concepts related to our work. Section 5.1.1 introduces the magnetic field, Section 5.1.2 describes the magnetic field sensor of the smartphones, and Section 5.1.3 elucidates dynamic time warping that serves as the similarity measure for our classifier.

5.1.1 Magnetic Field

The magnetic field at any point in space is represented by a vector quantity. It is specified by its magnitude as well as direction and measured in Tesla (T). In practice, magnetic fields are measured in the unit of millitesla (mT) or microtesla (μT). Electric current produces a magnetic field. The total magnetic field generated around an enclosed path is directly proportional to the current which passes through that path [181]. In standard computers, the CPU is one of the biggest consumers of the electricity. The power consumption of modern energy efficient CPUs is dynamically affected by the workload [134]. In the most fundamental case, overloading the CPU with computations will draw more current, and hence, will generate a stronger magnetic field. To demonstrate this effect, we created a script that recursively engages all available CPUs for 2 seconds and then sleeps for 2 seconds. Figure 5.1 shows the CPU usage and the generated magnetic field while executing this script.

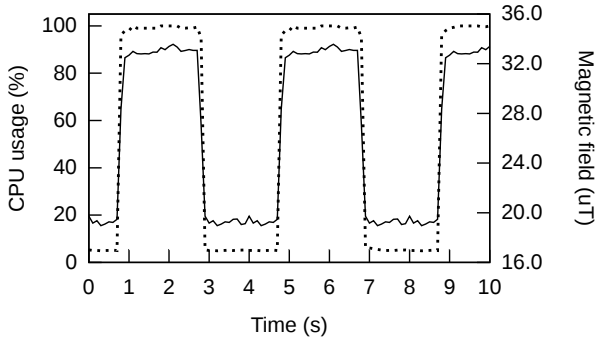


Figure 5.1: The CPU usage (dashed line) and the generated magnetic field (plain line) while executing a script that recursively uses all available CPUs for 2 seconds and then sleeps for 2 seconds

It is clear from Figure 5.1 that the magnetic field around a processor is directly affected by the workload of the processor. Hence, this side-channel can be exploited to infer a processor's activities.

5.1.2 Magnetic Field Sensor of the Smartphones

Nowadays, smartphones are equipped with a variety of sensors. One of these sensors is the magnetic field sensor, which typically measures the strength (magnitude and direction) of the magnetic field along the three physical (x, y, z) axes. Using the three dimensional data from the magnetic sensor, we can calculate the total magnetic field (M_{total}) as:

$$M_{total} = \sqrt{M_x^2 + M_y^2 + M_z^2}, \tag{5.1}$$

where M_x, M_y, and M_z represent the strength of the magnetic field along x, y, and z axis respectively.

5.1.3 Dynamic Time Warping

Dynamic Time Warping (DTW) is a widely adopted approach to find the optimal non-linear alignment between two time series that may vary in time as well as speed. DTW is also used as a distance measure to find similarity between time series [41]. Let us consider two discrete time series: $Q = (q_1, q_2, \ldots, q_i, \ldots, q_n)$ of length $n \in \mathbb{N}$ and $C = (c_1, c_2, \ldots, c_j, \ldots, c_m)$ of length $m \in \mathbb{N}$. DTW uses an $n \times m$ matrix, whose $(i,j)^{th}$ element represents the distance $d(q_i, c_j)$ between q_i and c_j (i.e., $d(q_i, c_j) = (q_i - c_j)^2$). Each matrix element (i,j) corresponds to the alignment between the points q_i and c_j. A warping path $W = (w_1, w_2, \ldots, w_k, \ldots, w_K)$ is a contiguous set of matrix elements that defines a mapping between Q and C. The k^{th} element of W is defined as $w_k = (i,j)_k$ for $max(m,n) \leq K < m + n - 1$. The warping path is typically subject to the following constraints:

1. Boundary condition: $w_1 = (1,1)$ and $w_K = (m,n)$.
2. Continuity: Given $w_k = (a,b)$, then $w_{k-1} = (a',b')$, where $a - a' \leq 1$ and $b - b' \leq 1$.
3. Monotonicity: Given $w_k = (a,b)$, then $w_{k-1} = (a',b')$, where $a - a' \geq 0$ and $b - b' \geq 0$.

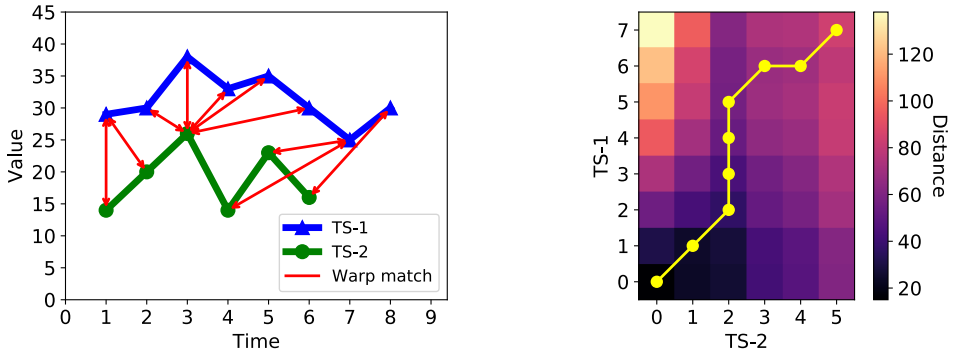
There could be several warping paths that satisfy the conditions reported above. However, our goal is to find the path that minimizes the warping cost:

$$DTW(Q,C) = min\left(\sqrt{\sum_{k=1}^{K} w_k}\right). \tag{5.2}$$

The optimal path can be found by computing the cumulative distance $\gamma(i,j)$ using the following recursive function:

$$\gamma(i,j) = d(q_i, c_j) + min(\gamma(i-1, j-1), \gamma(i-1, j), \gamma(i, j-1)). \tag{5.3}$$

Figure 5.2(a) shows the alignment between two discrete time series TS-1 and TS-2. The arrows indicate the points that are matched by the DTW algorithm. Figure 5.2(b) depicts the heat-matrix for these two time series. The color of a cell $(i,\ j)$ depicts the minimum distances to reach the cell $(i,\ j)$ from the cell $(0,\ 0)$. The optimal warping path has been highlighted using a yellow line that starts from the cell $(0,\ 0)$, which also satisfies boundary, continuity, and monotonicity constraints mentioned above.



(a) Alignment between TS-1 and TS-2

(b) Illustration of the optimal warping path

Figure 5.2: A representative example of DTW algorithm applied to two discrete time series: TS-1 and TS-2

Standard DTW is optimal, but it is computationally expensive, both in space and time complexity. Standard DTW has a complexity of $O(nm)$ where n and m are the lengths of the first and second time series respectively. The performance of DTW can be improved by restricting the amount of warping allowed, which typically limits the number of cells computed in the DTW distance matrix. There are several proposals that claim to improve DTW such as PrunedDTW [186], SparseDTW [35], FastDTW [183], and MultiscaleDTW [155,168]. These approaches [35,155,168,183,186] propose ways to speed-up exact DTW computation for two time series. Next, LB_Keogh lower bound [123] is a technique that allows efficient indexing and comparison of many time series via lower-bound approximation (less than exact DTW value). Moreover, Wang et al. [212] found that except for LB_Keogh lower bound [123], all other techniques are inefficient. Hence, in this work, we use LB_Keogh lower bound to speed up DTW, which is computed as shown in Eq. 5.4:

$$LB\_Keogh(Q,C) = \sqrt{\sum_{i=1}^{n} \begin{cases} (c_i - U_i)^2, & \text{if } c_i > U_i; \\ (c_i - L_i)^2, & \text{if } c_i < L_i; \\ 0, & otherwise. \end{cases}} \tag{5.4}$$

Here, L_i and U_i are lower and upper bounds for time series Q, which are defined as $L_i = min(q_{i-r} : q_{i+r})$ and $U_i = max(q_{i-r} : q_{i+r})$ for a reach r. LB_Keogh lower bound method has linear complexity, which makes it very useful for searching over large sets of time series.

5.2 Related Works

The side-channel information leakage has been thoroughly studied over the previous years. Several research works focused on various side-channels to leak information from different type of systems. We will primarily discuss previous studies that are relevant to our work.

Quisquater et al. [170] show that the electromagnetic attack on processors can reveal as much information as power consumption-based side-channel analysis. Mateos et al. [145] use specialized magnetic sensors to recover a secret key. Song et al. [188] investigate acoustic and magnetic side-channel attacks on 3D printers using smartphone's built-in sensors. Biedermann et al. [42] use the smartphone's magnetic sensors to fingerprint hard-drives. The authors are able to deduce ongoing system activities based on the emitted magnetic field due to movements of the hard drive's magnetic head. However, their methodology is inapplicable to the modern computers equipped with Solid State Drive (SSD) storage. Similarly, Matyunin et al. [146] establish a covert channel using the magnetic field between a laptop that is infected with a special program and a smartphone equipped with a magnetic sensor. The authors claim that their approach can decode the emanated signal up to a distance of roughly 12 cm from the laptop. ODINI [111] and MAGNETO [110] employ malware to control workload on the CPU, which, in turn, regulates the magnetic fields emitting from the target device. ODINI uses a high-precision magnetic sensor to receive data up to 150 cm while MAGNETO can receive data up to a maximum distance of 12 cm using a regular smartphone.

The literature on detecting the mining of cryptocurrencies is rather limited. Bonneau et al. [46] present a systematic study of various cryptocurrencies and discuss open research challenges. Huang et al. [118] in their analysis of Bitcoin mining malware have shown that modern botnets generate additional revenue via illicit mining. Eskandari et al. [99] present a security analysis of in-browser mining of cryptocurrencies. Recent works [135,140,173,180, 208] focus specifically on browser-based mining. However, only a small subset of cryptocurrencies supports in-browser mining. MineGuard [200] focuses on mining operations in the cloud infrastructure. The authors utilize privileged hardware performance counters to fingerprint and detect covert mining.

Our work is different from the state-of-the-art on the following dimensions: (1) our proposed approach is a generalized approach that applies to all forms of cryptomining on computers; (2) Our approach does not require any

specialized equipment, malware/software installation, user-/root-privileges, or even login-access to the monitored system. The only requirements are the physical proximity of the investigator and a magnetic sensor; (3) Our methodology does not impart any performance overhead on the monitored system; and (4) Our study includes cryptocurrencies supported by the top-10 mining pools, which covers the largest share of the market of cryptomining.

5.3 System Architecture

We explain the foundation of our work in Section 5.3.1, our data collection strategy in Section 5.3.2, our decision for selecting cryptocurrencies in Section 5.3.3, and the design of our classifier in Section 5.3.4.

5.3.1 Core Concept

The task of mining is to execute a core Proof-of-Work (PoW[2]) algorithm repeatedly, which means that a robust signature can be constructed for a particular algorithm. Interestingly, there are a limited number of PoW algorithms. Therefore, we focus on the mining algorithms. The major benefit of our strategy is that the signature constructed for an algorithm would be able to detect even metamorphic and polymorphic implementations of that algorithm used by other cryptocurrencies. Notably, such signature-based detection of cryptomining can be partially deceived by restricted mining. But, it would directly affect the hashing rate and consequently the profits; making the task of mining less appealing.

Previous works [42,110,111,146] have shown the effectiveness of the magnetic side-channel to exfiltrate information from computers. We designed our system for detecting and classifying covert cryptomining using the magnetic side-channel for the following reasons:

1. The examiner/investigator may not always have login-access[3] or root-privileges[4] on the device.

2. As discussed in Section 5.1.1, magnetic field emission is a fundamental phenomenon associated with electronic circuits and it can even penetrate air-gaps and a faraday-cage.

3. Since the miners have to adhere to and repeatedly execute the core PoW algorithm used by a cryptocurrency, the pattern of the magnetic field emitted while mining a cryptocurrency is distinct as well as consistent. For the same reason, even a smaller signature database is sufficient for reasonable classification results.

[2]We use "PoW" as the representative of various consensus algorithms.
[3]E.g., an administrator in BYOD culture, who suspects an employee's machine.
[4]E.g., an employee, who suspects an infection in the company-provided machine.

5.3.2 Dataset Collection

To better elucidate our work and to maintain the flow of reading, we first explain the data collection stage. We used two different systems in our experiments. These systems have the following configuration: (1) *S1*, a laptop with an Intel Core i5-7200U @ 2.50 GHz (1 socket x 2 cores x 2 threads = 4 logical compute resources) processor, 8 GB memory, 256 GB SSD storage, and Intel HD Graphics 620 mounted on Dell Inc. 0M60Y2 motherboard with Ubuntu 16.04 as the operating system and (2) *S2*, a laptop with an Intel Core i7-8550U @ 1.80 GHz (1 socket x 2 cores x 4 threads = 8 logical compute resources) processor, 16 GB memory, 512 GB SSD storage, and Intel UHD Graphics 620 mounted on Dell Inc. 02PG84 motherboard with Ubuntu 18.04 as the operating system. To show the effectiveness of the proposed method, we used an ordinary smartphone (Samsung Galaxy S5 running Android 6.0.1), hereinafter referred to as *probe* device, to record the generated magnetic field. Figure 5.3 depicts a representative demonstration of data collection on *S1* using the probe device. Section 5.5.2 presents a detailed discussion on probe's orientation and position.

Figure 5.3: A representative demonstration of data collection on *S1* using the probe device

We mined and profiled each cryptocurrency (discussed in Section 5.3.3) individually and collected a total of thirty samples per cryptocurrency per system, where each sample comprises measurements taken over a time interval of thirty seconds. The sampling rate of probe's magnetic sensor was 10Hz. To obtain clean signatures of the core PoW algorithms, we profiled the miners in their stable stage, i.e., omitting the bootstrapping phase. As representatives of non-mining tasks (negative-class), we chose: Internet browsing; PDF document scrolling; *Skype* test call; 3D benchmarking; solving N-queens problem (N=18); 4K video streaming; H.264 video encoding; network performance test via *iperf* tool; machine learning with *scikit-learn*; deep learning with *TensorFlow*; *stress-ng* [6] stress test via CPU-only workers; and *stress-*

ng stress test with CPU, memory, I/O, and disk workers together. It is worth mentioning that these user-tasks represent low to high compute-intensive tasks and all belong to the same category (i.e., non-mining) for classification purposes. Similar to the mining tasks, we profiled each non-mining task for the same time interval and number of samples.

Before any experiment was performed, we estimated the background magnetic field to calibrate subsequent measurements (triplets). We took 100 measurements (10sec @ 10Hz) along the three (x, y, z) axes and calculated the mean ($\bar{M_x}$, $\bar{M_y}$, $\bar{M_z}$) of the background noise. We calibrated each measurement, collected during the experiments, by eliminating the mean background noise from it. Finally, we computed M_{total} for each calibrated measurement using Eq. 5.1.

5.3.3 Cryptocurrencies & Miners

In the context of cryptocurrency mining, miners pool their resources so they can generate blocks more quickly, and therefore earn a portion of the block reward on a consistent basis. Mining pools are characterized by their hashing power. Table 5.1 lists the cryptocurrencies supported by the top-10 mining pools [17], which collectively comprise the largest share (84% during Q3 2018) of the cryptomining market. See Appendix A for acronyms and their corresponding cryptocurrency. We included all the cryptocurrencies supported by these mining pools in our experiments. Additionally, we included QRK whose mining algorithm - unlike other currencies - consists of a combination of different hashing algorithms. To mine these currencies, we used open-source miner programs that are readily available online. Each miner was configured to mine with public mining pools and to use all available CPUs on the machine. Table 5.2 lists the mining algorithm of different cryptocurrencies as well as the CPU miners that we used to mine them.

As explained in Section 5.3.1, different cryptocurrencies that employ the same mining algorithm exhibit the same signature. Therefore, it is sufficient to consider one currency for each mining algorithm listed in Table 5.2. We excluded BCH, SBTC, UBTC, ETC, and XMC. In our study, we used only CPU-based miners as the proof-of-concept implementation. Nevertheless, our approach is also valid to distinguish GPU-based miners because a GPU operates differently than a CPU. For the same reason, GPU-based mining generates a distinct magnetic field in terms of magnitude as well as form. As a representative example, Figure 5.4 shows the generated magnetic field while mining XMR on CPU and on GPU[5].

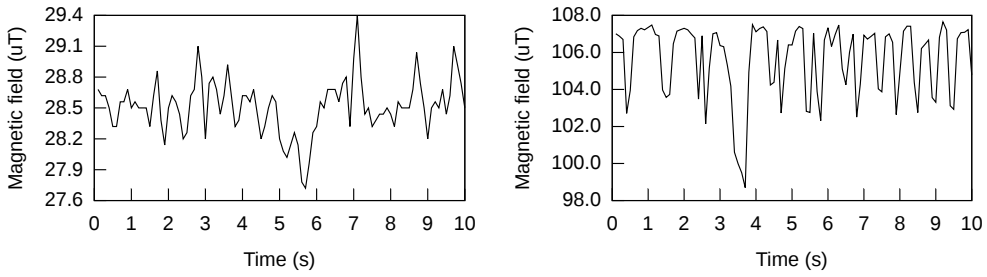
[5]Using ccminer v2.3 on 4 GB NVIDIA GeForce GTX 960M.

N.	Mining pool	Cryptocurrency													
		BCD	**BCH**	**BTC**	**BTM**	**DASH**	**DCR**	**ETC**	**ETH**	**LTC**	**SBTC**	**SC**	**UBTC**	**XMC**	**XMR**	**XZC**	**ZEC**
1	BTC.com	✗	✓	✓	✗	✗	✗	✗	✗	✗	✓	✗	✓	✗	✗	✗	✗
2	AntPool	✗	✓	✓	✓	✓	✗	✓	✓	✓	✗	✓	✗	✓	✗	✗	✓
3	ViaBTC	✗	✓	✓	✗	✓	✗	✓	✓	✓	✗	✗	✗	✗	✗	✗	✓
4	SlushPool	✗	✗	✓	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✓
5	F2Pool	✗	✗	✓	✗	✓	✓	✓	✓	✓	✗	✓	✗	✓	✓	✓	✓
6	BTC.top	✗	✓	✓	✗	✓	✗	✗	✗	✓	✗	✗	✗	✗	✗	✗	✗
7	Bitclub.network	✗	✗	✓	✗	✓	✗	✓	✓	✗	✗	✗	✗	✗	✓	✗	✓
8	BTCC	✓	✓	✓	✓	✗	✗	✗	✗	✓	✓	✗	✗	✗	✗	✗	✗
9	BitFury	✗	✗	✓	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗
10	BW.com	✗	✗	✓	✗	✗	✗	✓	✓	✓	✗	✓	✓	✗	✗	✗	✗

Table 5.1: Cryptocurrencies supported by the top-10 mining pools

Cryptocurrency	Mining algorithm	CPU miner
BCD	X13	cpuminer-opt 3.8.8.1
BCH, BTC, SBTC, UBTC	SHA-256	cpuminer-multi 1.3.4
BTM	Tensority	bytom-wallet-desktop 1.0.2
DASH	X11	cpuminer-multi 1.3.4
DCR	Blake256-r14	cpuminer-multi 1.3.4
ETC, ETH	Ethash (Modified Dagger-Hashimoto)	geth 1.7.3
LTC	scrypt	cpuminer-multi 1.3.4
QRK	BLAKE + Grøstl + Blue Midnight Wish + JH + Keccak (SHA-3) + Skein	cpuminer-multi 1.3.4
SC	BLAKE2b	gominer 0.6
XMC, XMR	CryptoNight	cpuminer-multi 1.3.4
XZC	Lyra2z	cpuminer-opt 3.8.8.1
ZEC	Equihash	Nicehash nheqminer 0.3a

Table 5.2: Mining algorithm and CPU miner for different cryptocurrencies



(a) On CPU (b) On GPU

Figure 5.4: Generated magnetic field while mining XMR

5.3.4 Classifier Design

The data for the generated magnetic field can be represented as a time series. Thus, our problem converts to time series classification. In our scenario, we can identify the following two main objectives of the classification stage:

- Classify if a given instance represents the mining activity or not;
- If so, then predict the specific currency (algorithm).

Now, we discuss our data preprocessing, machine learning model selection, training, and prediction phase.

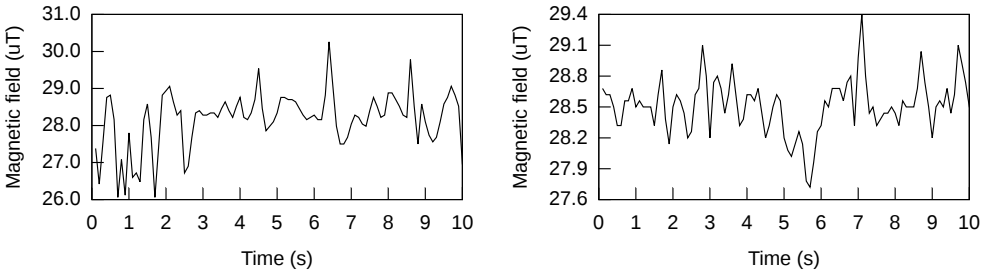
Data preprocessing

The results of time series classification are affected by the quality of input data. All the time series data in our dataset are of equal length. Before

starting the training, we employ a scaling function to normalize the input data. In particular, following the suggestion from the work [171], we use the Z-normalization technique. See Appendix B for further details on the scaling technique. Next, we *smooth* [15] the input data to remove noise.

Machine learning

Figure 5.5 depicts the generated magnetic field while mining BTC and XMR. The time series graphs shown in Figure 5.5 are different from each other because each of these cryptocurrencies uses a distinct PoW algorithm that performs a discrete task and has a different iteration cycle.



(a) BTC (b) XMR

Figure 5.5: A representative example of the generated magnetic field while mining BTC and XMR

We use the K-Nearest Neighbors (KNN) algorithm for the classification of our time series consisting of the values for M_{total} at each measurement, where DTW distance serves as the similarity measure. In particular, we use the KNN classifier with K = 1 because previous studies on the classification of time series data demonstrated that DTW-based 1NN classifier - which selects the first nearest neighbors - is "the best" [98, 152], "Nearest Neighbor DTW is very hard to beat" [39], and "1NN with DTW is exceptionally hard to beat" [214]. Nevertheless, we also performed stratified 4-fold cross-validation on training data, which we obtain from 80-20% stratified partitioning of our dataset into training-test split, and we observed least error rate for K=1 among all the single digit odd values of K.

Training and prediction

Our classification model is instance-based. For every instance in the test-set, a search is performed through all the instances in the training-set to find the most similar time series. Given the quadratic complexity of DTW, we use LB_Keogh lower bound (see Section 5.1.3) to speed up the classification stage. Given a new instance to classify, the prediction is made for both the objectives discussed at the beginning of this section.

5.4 Evaluation

Here, we describe the evaluation procedure used to thoroughly assess the quality of our proposed approach. For the objectives of the classification stage, mentioned in Section 5.3.4, we performed the following five experiments: (1) *binary* classification; (2) *currency* classification; (3) *nested* classification; (4) *unseen-miner program* classification; and (5) *cross-platform* classification. In order to increase the statistical significance of the results, we repeated each experiment five times with stratified 80-20% training-test partitioning. It is worth to state that even though the dataset has been collected in a controlled setup, our experiments fairly simulate the real-world scenario, where samples are gathered in real-time. Table 5.3 describes the sample distribution in our dataset for each system, i.e., *S1* and *S2*. Here, sub-classes of the mining task refer to the cryptocurrencies (discussed in Section 5.3.3) while sub-classes of the non-mining task refer to the actual user-tasks that belong to the negative class (mentioned in Section 5.3.2). It is important to mention that we created a single training-set where we kept the instances from both *S1* and *S2* together.

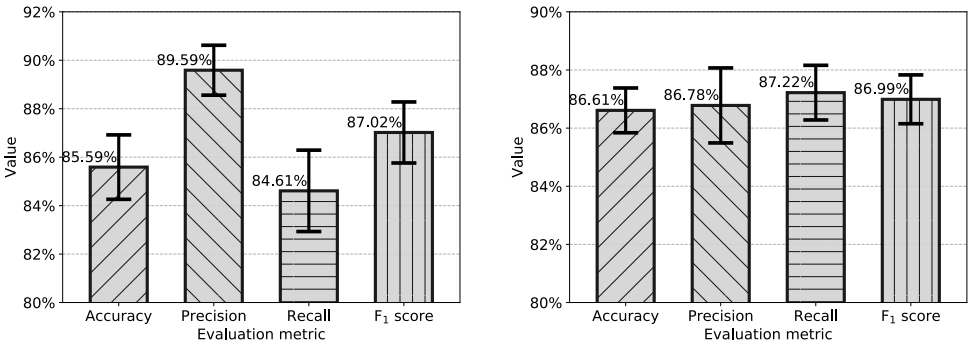
Task	**Sub-classes per task**	**Samples per sub-class**	**Total samples per task**
Mining	12	30	360
Non-mining	12	30	360

Table 5.3: Dataset: name of the task, sub-classes per task, samples per sub-class, and total samples per task for each system.

We evaluated our classifier using standard classification metrics: Accuracy, Precision, Recall, and F_1 score. For the statistical certitude of our results, we report the mean and the margin of error for the results with 95% confidence interval from five runs of each experiment for each of the evaluation metric. See Appendix B for details on the evaluation metrics and the related statistical terms. We use the notation $mean \pm margin\_of\_error$ to report our results.

5.4.1 Binary Classification

In this setting, we consider our classification problem as a binary classification task for Mining (positive) class and non-mining (negative) class. All the instances of various cryptocurrencies are treated as the positive-class while all the instances of non-mining user-tasks fall in the negative-class. This assessment aims to evaluate our classifier's ability to detect the presence of cryptomining activities. Figure 5.6 presents the results of *binary* classification. Figure 5.6 (a) and Figure 5.6 (b) correspond to *S1* and *S2*, respectively.



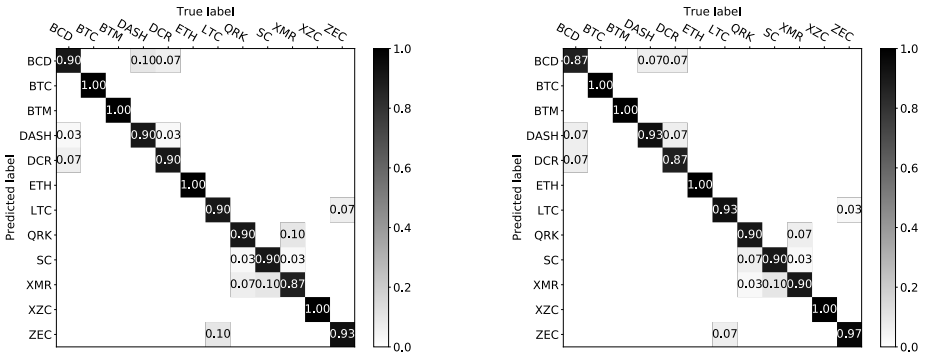
(a) *S1* (b) *S2*

Figure 5.6: Results of the *binary* classification (whiskers represent the margin of error)

On *S1*, we achieved an average accuracy of 85.59%±1.33%, precision of 89.59%±1.03%, recall of 84.61%±1.68%, and F_1 score of 87.02%±1.26% while on *S2*, we attained an average accuracy of 86.61%±0.77%, precision of 86.78%±1.29%, recall of 87.22%±0.94%, and F_1 score of 86.99%±0.84%.

5.4.2 Currency Classification

We designed this experiment to comprehend the level of difficulty in distinguishing various cryptocurrencies. Hence, the input data for this experiment comprised of instances belonging only to cryptocurrencies. Figure 5.7 showcases the confusion matrix for classification among various cryptocurrencies. We drew the confusion matrices using the aggregate results from all five runs. Figure 5.7 (a) and Figure 5.7 (b) correspond to *S1* and *S2*, respectively.



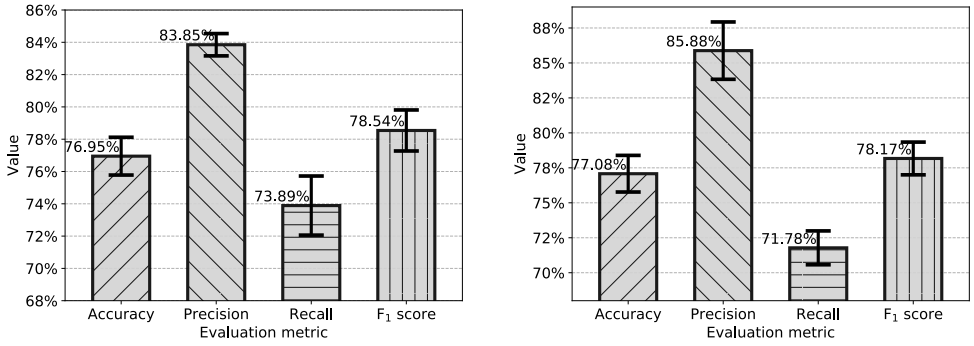
(a) *S1* (b) *S2*

Figure 5.7: Confusion matrix for *currency* classification

On both *S1* and *S2*, our classifier achieved an overall-average performance of over 93% for each of the evaluation metric. Furthermore, the results from this assessment also help us to better understand the outcomes of *nested* classification, which is discussed next.

5.4.3 Nested Classification

This assessment aims to evaluate our classifier's ability to fulfill both our classification objectives, i.e., first, identify whether the given instance represents a mining activity, and if so, then predict the specific currency. It is worth to mention that an error in the primary stage of the nested classification influences the subsequent stage. Furthermore, given that our classifier makes a correct decision in the primary stage, the difficulty level of the subsequent stage affects the final results. Figure 5.8 depicts the results of the *nested* classification. Figure 5.8 (a) and Figure 5.8 (b) correspond to *S1* and *S2*, respectively.



(a) *S1* (b) *S2*

Figure 5.8: Results of the *nested* classification (whiskers represent the margin of error)

On *S1*, we attained an average accuracy of 76.95%±1.17%, precision of 83.85%±0.69%, recall of 73.89%±1.83%, and F_1 score of 78.54%±1.27% while on *S2*, we achieved an average accuracy of 77.08%±1.31%, precision of 85.88%±2.05%, recall of 71.78%±1.21%, and F_1 score of 78.17%±1.17%. Given the lenient requirements (mentioned in Section 5.2) of our methodology, we believe that the results of the *nested* classification are justifiable. Nevertheless, our primary aim is to identify the presence of the covert cryptomining, for which, our *binary* classification has shown promising results.

5.4.4 Unseen-miner Program Classification

Since there can be more than one miner programs for a cryptocurrency and training a classifier on every miner program might not be possible. There-

fore, we designed this experiment to evaluate the proficiency of our approach in such circumstances. The goal of this experiment was to perform the *binary* classification of all mining and non-mining samples, as mentioned in Section 5.4.1. However, we selected two additional miner programs for BTC, namely, BFGMiner 5.5 and cgminer 4.10. During the training, the classifier was exposed to samples from one of the three miner-programs for BTC. In contrast, during the testing phase, samples from one of the remaining two miner-programs for BTC were used. Table 5.4 reports our results of classifying samples from the miner programs that were not seen in training.

System	Set	Accuracy	Precision	Recall	$\mathbf{F_1}$ score
S1	α_β	0.966 ± 0.015	0.969 ± 0.015	0.965 ± 0.016	0.967 ± 0.016
	α_γ	0.952 ± 0.024	0.957 ± 0.022	0.951 ± 0.023	0.953 ± 0.024
	β_α	0.966 ± 0.016	0.968 ± 0.016	0.964 ± 0.016	0.966 ± 0.016
	β_γ	0.969 ± 0.016	0.971 ± 0.015	0.967 ± 0.016	0.969 ± 0.016
	γ_α	0.955 ± 0.023	0.958 ± 0.020	0.954 ± 0.022	0.954 ± 0.022
	γ_β	0.966 ± 0.021	0.970 ± 0.019	0.964 ± 0.021	0.966 ± 0.021
S2	α_β	0.957 ± 0.010	0.961 ± 0.011	0.955 ± 0.010	0.957 ± 0.011
	α_γ	0.943 ± 0.018	0.949 ± 0.017	0.941 ± 0.016	0.943 ± 0.018
	β_α	0.951 ± 0.014	0.954 ± 0.014	0.950 ± 0.012	0.952 ± 0.013
	β_γ	0.954 ± 0.015	0.957 ± 0.015	0.953 ± 0.014	0.955 ± 0.015
	γ_α	0.941 ± 0.016	0.945 ± 0.015	0.941 ± 0.015	0.941 ± 0.015
	γ_β	0.953 ± 0.019	0.957 ± 0.017	0.951 ± 0.018	0.953 ± 0.019

Table 5.4: Results of the *unseen-miner program* classification

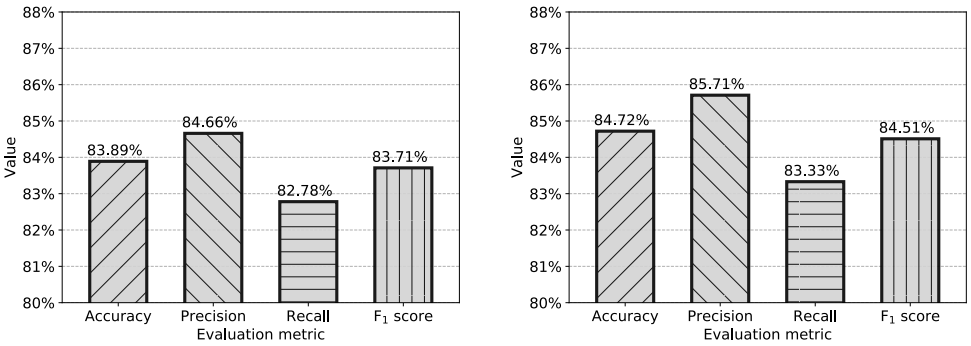
The notation M_N means that for BTC, the classifier was trained with samples from M while samples from N were used for testing. α =cpuminer-multi 1.3.4, β = BFGMiner 5.5, and γ = cgminer 4.10. It is important to mention that even though we performed the classification with all the mining and non-mining sub-classes, Table 5.4 presents the results only for BTC mining to preserve the goal of this experiment.

As mentioned in Section 5.3.1, the miners have to adhere to the core PoW algorithm used by a cryptocurrency. Our results presented in Table 5.4 support our notion that the pattern of the magnetic field emitted while mining a given cryptocurrency is consistent across different miner programs.

5.4.5 Cross-platform Classification

We designed this experiment considering one of our key motivations, i.e., to build a system that can detect covert cryptomining in situations where the hardware is heterogeneous, e.g., BYOD workplace. Here, we used two additional laptops, *S1'* and *S2'*, to collect a new test set. *S1'* and *S2'* has a distinct hardware configuration but identical processor as *S1* and *S2*, respectively. For each sub-class of both mining and non-mining tasks, we collected

15 separate samples on both *S1'* and *S2'*. The target of this experiment was to perform the *binary* classification of mining and non-mining samples, as mentioned in Section 5.4.1. We used our dataset collected previously on *S1* and *S2* as the training set, but for testing, we used samples obtained from *S1'* and *S2'*. Figure 5.9 depicts the results of *cross-platform* classification. Figure 5.9 (a) and Figure 5.9 (b) correspond to *S1'* and *S2'*, respectively.



(a) *S1'* (b) *S2'*

Figure 5.9: Results of the *cross-platform* classification

Essentially, our proposed approach is to profile the magnetic field emission of a processor for the set of available mining algorithms. As shown in Figure 5.9, the performance of the *binary* classifier on *S1'* and *S2'* is nearly at par (with a maximum performance degradation of 5.5%, which is in the precision) with its average performance on *S1* and *S2* (see Figure 5.6), where the training set was originally collected.

Finally, we used the profile of one processor to classify samples from another processor. In these separate experiments, we used training-set from one device (*S1* | *S2*) and test-set from another device (*S2* or *S2'* | *S1* or *S1'*). We found that the profile of one processor may not be reliably used to classify instances from another processor. In fact, these results align with our fundamental idea to profile the magnetic field emission of individual processors for the set of available mining algorithms.

5.5 Discussion

In this section, we discuss the important aspects of our proposed approach and address its potential limitations.

5.5.1 Zero-day Attack

In our context, a zero-day attack would be to mine a cryptocurrency that uses a brand new or custom PoW algorithm. However, for such a currency

to have a value/worth in the real-world, its PoW algorithm must be mathematically robust as well as accepted by the crypto-community and its core network must be supported by moderate-to-large scale miners/pools. Hence, by the time a zero-day cryptocurrency becomes ready for mining, its algorithm would be public knowledge, providing us sufficient time to train our system for this new currency's signature.

On another side, conscious-miners as well as the actors behind unconscious-mining tend to mine more profitable currencies (whose mining algorithms are certainly public knowledge) to maximize their profit and avoid hashing the less valuable ones. In our experiments, we considered all the mining algorithms supported by the top-10 mining pools, which indeed are the most mined cryptocurrencies.

5.5.2 Probe's Orientation & Position

The orientation as well as the position of the probe with respect to the processor are the critical aspects of our work. Our approach relies on the total magnetic field (M_{total}), which is computed using Eq. 5.1. The magnetic sensor's reading - which can be positive as well as negative depending on the direction - for each component of the magnetic field is squared first, which eliminates the influence of the probe's orientation.

Since the magnetic fields decay over distance, the distance (position) of the probe from the processor can be seen as a limitation of our approach. In our scenario, the investigator has at least physical-access - if not login-access - to the system. Hence, one can place the probe near the processor simply by understanding the system's physical architecture. Any light-offset in positioning the probe would still perceive the same waveform of the generated magnetic field though with a different amplitude, which is neutralized during data normalization phase. Nevertheless, using high-precision magnetic sensors may help to manage this limitation to some extent.

5.5.3 Interference due to Other Processes

The miner programs tend to exploit all available compute resources and deprive other processes of these resources. However, minute interference due to the occasional scheduling of other processes can be handled by the very nature of our classification methodology, DTW in particular. Such interference would be minimum in the case of the conscious-miners, who would allocate all the resources to the mining process to maximize the profit. Whereas unconscious-mining would interfere with its victim's tasks; this is the situation[6] where the victims can halt their tasks and use our system to detect covert cryptomining.

[6]Modern malware can hide and circumvent standard detection approaches.

5.5.4 Scalability

The fundamental idea of our proposed approach is to profile the magnetic field emission of a processor for the set of available mining algorithms. Given the finite number of CPUs/GPUs, obtaining signatures is merely a data collection task. At the beginning, it might appear a tedious task. But, once completed, keeping it up-to-date is relatively easy because only a limited number of processors are released from time to time.

5.5.5 Restricted Mining

A mining strategy to evade detection from our proposed methodology can be *restricted mining* that aims to change the pattern of the emitted magnetic field. Here, the miner can either throttle the mining down or perform arbitrary tasks during mining. But, both maneuvers would directly affect the hashing rate and consequently the profits; making the task of mining less appealing. Nevertheless, like any signature-based detection technique, it may be seen as a limitation of our work.

5.6 Summary

In this chapter, we present a novel methodology to detect and classify covert cryptomining. Our proposed approach focuses on the core of the cryptomining, i.e., mining algorithms. Since it uses the magnetic side-channel, it works even if the examiner does not have login-access or root-privileges on the suspect machine. In our study, we considered a wide variety of cryptocurrencies and empirically demonstrated the effectiveness of our system.

Chapter 6

Detecting Covert Miners via Hardware Performance Counters

In continuation of our work presented in Chapter 5, we propose an approach that is designed for another critical and real-world scenario of covert cryptomining. In the former case, we utilize magnetic side-channel to detect two types of covert cryptomining, i.e., conscious-mining and unconscious-mining. In that case, we assume that the investigator does not have login-access[1] (corresponds to the case of conscious-mining) or root-privileges[2] (corresponds to the case of unconscious-mining) on the suspect device.

In this chapter, we specifically focus on unconscious-mining. However, unlike the previous case, the investigator here owns the device and can elevate his/her privileges to the root-level. This category of covert cryptomining is also known as cryptojacking. Similar to covert cryptomining, cryptojacking is also defined as an unauthorized use of the computing resources on computers, tablets, mobile phones, etc. to mine cryptocurrencies.

Cybercriminals have made several ingenious attempts to spread cryptojackers in the form of malware [31], malicious browser extensions [20], etc. by exploiting vulnerability [27], compromising third-party plug-ins [29], maneuvering misconfigurations [19], taking advantage of web-based hosting service [21], and so on. To evade intrinsic detection techniques (e.g., processor's usage), some cryptojackers suspend their execution when the victim is using the computer [81], use "pop-under" windows to keep mining for a comparatively longer duration [14], and utilize legitimate processes of the operating system to mine [57]. Moreover, merely monitoring CPU load, etc. is an ineffective strategy because of both false positives and false negatives [135]. To further aggravate the situation, various cryptocurrency mining services (e.g., Coinhive [1], Crypto-Loot [3]) easily integrate into web-

[1] E.g., an administrator in BYOD culture, who suspects an employee's machine.

[2] E.g., an employee, who suspects an infection in the company-provided machine.

sites to monetize the computational power of their visitors. In fact, cryptojacking attacks exceeded ransomware attacks in 2018 and affected five times more systems as compared to ransomware [33].

To tackle such a complex scenario, we focus on the core mining algorithms and utilize Hardware Performance Counters (HPC) to create clean signatures that grasp the execution pattern of these algorithms on a processor. HPC are special-purpose registers in modern microprocessors that count and store hardware-related activities. These activities are commonly referred to as hardware *events*[3]. HPC are often used to conduct low-level performance analysis and tuning. HPC-based monitoring has very low-performance overhead, which makes it suitable even for latency-sensitive systems. Several works have shown the effectiveness of using HPC to detect generic malware [85, 209, 216], kernel-level rootkits [210], side-channel attacks [55], unauthorized firmware modifications [211], etc.

Contributions: The major contributions of our work are as follows:

1. We propose an efficient approach to detect cryptojacking. In particular, our approach uses HPC to profile the core of the mining process, i.e., the mining algorithms, on a given processor to accurately identify cryptomining in real-time. We designed our solution to be a generic one, i.e., it is not tailored to a particular cryptocurrency or a specific form (e.g., browser-based) of cryptomining.

2. We exhaustively assess the quality of our proposed approach. To this end, we designed six different experiments: (1) *binary* classification; (2) *currency* classification; (3) *nested* classification; (4) *sample length*; (5) *feature relevance*; and (6) *unseen-miner program* classification. For a thorough evaluation of our proposed solution, we considered all the cryptocurrencies supported by the top-10 mining pools. Our results show that our classifier can accurately classify cryptomining activities.

3. In the spirit of reproducible research, we make our collected datasets and the code publicly available[4].

Organization: The remainder of this chapter is organized as follows: We explain our system's architecture in Section 6.1 and discuss its evaluation in Section 6.2. Section 6.3 addresses the potential limitations of our proposed solution. Finally, Section 6.4 summarizes the chapter.

[3]Formally, an *event* is defined as a countable activity, action, or occurrence on a device.
[4]https://tinyurl.com/y2uq25y3

6.1 System Architecture

We elucidate the key concept behind our approach in Section 6.1.1, our data collection phase in Section 6.1.2, selection of cryptocurrencies Section 6.1.3, and our classifier's design in Section 6.1.4.

6.1.1 Core Concept

The task of cryptomining requires a miner to run the core PoW[5] algorithm repetitively to solve the cryptographic puzzle. At a coarse-grained level, some PoW algorithms are processor-oriented (e.g., BTC) while some are memory-oriented (e.g., XMR) due to their underlying design. At a fine-grained level, each PoW algorithm has its own unique mathematical/logical computations (or, in other words, the sequence of operations). Thus, each algorithm upon execution affects some specific *events* more as compared to other *events* on the processor. Consequently, when an algorithm is executed several times repetitively, the "more" affected *events* outnumber the other - relatively under affected - *events*. It means that a discernible signature can be built using the relevant *events* for a PoW algorithm. As a representative example, Figure 6.1 depicts the variation in *events* while mining different cryptocurrencies and performing some common user-tasks. LTC, for instance, shows a more erratic pattern in cache-misses as compared to the other *events* that are affected during LTC mining. On the other hand, a Skype video call has more disparity in context-switches.

As discussed in Chapter 5, there is a finite number of PoW algorithms upon which the cryptocurrencies are established. Thus, we again concentrate on the mining algorithms instead of individual currency in our solution. The primary advantage of our approach is that the signature built for an algorithm would be able to identify even polymorphic, metamorphic, and heavily obfuscated implementations of that algorithm because the core PoW algorithm - that we profile in our solution - remains the same. To this end, we use supervised machine learning (explained in Section 6.1.4) to construct signatures and build our classifier.

On another side, an adversary may attempt to circumvent such signature-based detection in the following ways: (1) by controlling/limiting the mining; or (2) by neutralizing the signatures. Limiting the mining would reduce the hashing rate, which would indeed make the mining less profitable. Whereas, to neutralize the signatures, the adversary has to succeed in two main hurdles. First, the adversary must have to find those computation(s) that only changes those *events* that are unrelated to the PoW algorithm. Second, the adversary must have to run these computation(s) in parallel to the PoW algorithm, which would again hamper the hashing rate, and thus the profit.

[5]We use the term "PoW" to represent different consensus algorithms.

(a) Number of L1-dcache-loads

(b) Number of L1-dcache-loads-misses

(c) Number of instructions

(d) Number of dTLB-loads

(e) Number of cache-misses

(f) Number of context-switches

Figure 6.1: A representative example of variation in *events* while mining different cryptocurrencies and performing some common user-tasks. HPC were polled every 100ms. The line-points in the graphs do not represent data points and are merely used to make lines distinguishable.

6.1.2 Data Collection

To better explain our work, we first describe what data we collect and how we collect it. We used the *perf* [5] tool to profile the processor's *events* using HPC. In particular, we focus on hardware[6] *events* (e.g., branch-misses), software[7] *events* (e.g., page-faults), and hardware cache[8] *events* (e.g., cache-misses) on CPU as the mining processes - depending on their design - require

[6]Basic events, measured by Performance Monitoring Units (PMU).
[7]Measurable by kernel counters.
[8]Data- and instruction-cache hardware events.

different type of resources. We profiled each program of both positive (mining) and negative (non-mining) class individually and collected a total 50 samples per program. Each sample consists of recordings of 28 *events* (described in Table 6.1) for 30 seconds with a sampling rate of 10Hz, which means that each sample comprises 300 readings of 28 *events*, i.e., 8400 readings. To obtain clean signatures: (1) we profiled each program in its stable stage, i.e., omitting the bootstrapping phase and (2) restarted the system to remove any trace of the previous sample.

For the positive class, we profiled a total of 11 cryptocurrencies discussed in Section 6.1.3. As the representatives of negative class, we chose: 3D rendering; *7z* archive extraction of *tar.gz* files; H.264 video encoding of raw video; solving *mqueens* problem; Nanoscale Molecular Dynamics (NAMD) simulation; *Netflix* movie playback; execution of Random Forest (RF) machine learning algorithm; *Skype* video calls; *stress-ng* [6] stress test with CPU, memory, I/O, and disk workers together; playing *Team Fortress 2* game; and Visual Molecular Dynamics (VMD) modeling and visualization. It is worth mentioning that these user-tasks represent low to high resource-intensive tasks.

We used two different systems to build our dataset for the experiments. The configuration of these systems are as follows: (1) *S1*, a laptop with an Intel Core i7-7500U @ 2.70 GHz (1 socket x 2 cores x 2 threads = 4 logical compute resources) processor, 8 GB memory, 512 GB SSD storage, NVIDIA GeForce 940MX 2 GB dedicated graphic card, Linux kernel 4.14 and (2) *S2*, a laptop with an Intel Core i7-8550U @ 1.80 GHz (1 socket x 2 cores x 4 threads = 8 logical compute resources) processor, 16 GB memory, 512 GB SSD storage, Linux kernel 4.14.

All miner programs and the *perf* tool were launched in *user*-mode. Even though we did not use any system-level privileges, we believe that using *root* permissions for defense against cryptojacking is reasonable. It is worth emphasizing that even though the dataset has been accumulated in a controlled setup, our experiments (discussed in Section 6.2) well simulate real-world scenario, where samples are collected in the real-time.

6.1.3 Cryptocurrencies & miners

Typically, the mining pools are characterized by their hashing power. Table 5.1 shows the top-10 mining pools [17] and the cryptocurrencies mined by them. At the time writing, these ten mining pools collectively constitutes the biggest share (84% during Q3 2018) of the cryptomining business. See Appendix A for the acronyms and their corresponding cryptocurrency.

We considered all the cryptocurrencies mentioned in the Table 5.1 in our experiments. We used open-source miner programs to mine these cryptocurrencies. Each miner program was configured to mine with public mining pools and to utilize all available the CPUs present on the system. At the

Event	Type	Description
branch-instructions	HW	N. of retired branch instructions.
branch-load-misses	HW	N. of Branch load misses.
branch-loads	HW	N. of Branch load accesses.
branch-misses	HW	N. of mispredicted branch instructions.
bus-cycles	HW	N. of bus cycles, which can be different from total cycles.
cache-misses	HC	N. of cache misses.
cache-references	HC	N. of cache accesses.
context-switches	SW	N. of context switches.
cpu-migrations	SW	N. of times the process has migrated.
dTLB-load-misses	HC	N. of load misses at data TLB.
dTLB-loads	HC	N. of load hits at data TLB.
dTLB-store-misses	HC	N. of store misses at data TLB.
dTLB-stores	HC	N. of store hits at data TLB.
instructions	HW	N. of retired instructions.
iTLB-load-misses	HC	N. of instruction fetches that missed instruction TLB.
iTLB-loads	HC	N. of instruction fetches that queried instruction TLB.
L1-dcache-load-misses	HC	N. of load misses at L1 data cache.
L1-dcache-loads	HC	N. of loads at L1 data cache.
L1-dcache-stores	HC	N. of stores at L1 data cache.
LLC-load-misses	HC	N. of load misses at the last level cache.
LLC-loads	HC	N. of loads at the last level cache.
LLC-store-misses	HC	N. of store misses at the last level cache.
LLC-stores	HC	N. of stores at the last level cache.
mem-loads	HC	N. of memory loads.
mem-stores	HC	N. of memory stores.
node-load-misses	HC	N. of load hits at Non-Uniform Memory Access (NUMA) node.
node-loads	HC	N. of load misses at NUMA node.
node-store-misses	HC	N. of store hits at NUMA node.
node-stores	HC	N. of store misses at NUMA node.
page-faults	SW	N. of page faults.
ref-cycles	HW	N. of total cycles; not affected by CPU frequency scaling.
task-clock	SW	The clock count specific to the task that is running.

Table 6.1: The *events* that we monitor using HPC. Here, HW = hardware, SW = software, and HC = hardware cache *event*.

time of our experiments, the miner program for SC was not able to mine using only CPU. Hence, we excluded SC from our experiments. To compensate SC, we included QRK whose mining algorithm - in contrast to other cryptocurrencies - uses multiple hashing algorithms. Table 5.2 shows the mining algorithm of different cryptocurrencies and the CPU miners we used.

Since our approach focuses on the underlying core PoW algorithm, we considered one currency for every mining algorithm mentioned in Table 5.2. We excluded BCH, SBTC, UBTC, ETC, and XMC in our study. As the proof-of-concept implementation, we considered only CPU-based miner programs because each computer has at least one CPU, which cryptojackers can harness to mine. Nevertheless, our approach is also valid to distinguish GPU-based miners because dedicated profiling tools, such as the *nvprof* [4] tool for NVIDIA GPUs, allow us to monitor GPU *events*. Apart from most of the standard *events* found on CPUs, GPUs have several dedicated *events* that can assist in creating unique signatures for GPUs.

6.1.4 Classifier Design

In this section, we elucidate the design of our classification methodology. Algorithm 2 describes the pipeline of our classifier.

Algorithm 2 Pseudo code for our supervised classification.

1: **for** each run i from 1 to 10 **do**
2: Create *raw_train_set* and *raw_test_set* by 90-10% stratified partitioning.
3: *Data preprocessing*
• Replace *NaN* values from *raw_train_set* and *raw_test_set* with arithmetic mean of the considered event.
4: *Feature engineering*
• *train_set* := Extract_feature(*raw_train_set*)
• *test_set* := Extract_feature(*raw_test_set*)
5: *Feature scaling*
• scaler := StandardScaler()
• scaler.fit(*train_set*) ▷*Fit scaler on train_set*
• scaler.transform(*train_set*)
• scaler.transform(*test_set*)
6: *Feature selection*
• Compute features' importance with *forests of trees* on *train_set* and select the most relevant features.
7: *Training*
• Learn the model parameters for the given classifier (RF/SVM) on the training set using grid search with 5-fold stratified CV.
8: *Predict/classify* the *test_set*.
9: **end for**

Our supervised classification algorithm begins with splitting the base-dataset of 1100 samples (2 classes x 11 instances x 50 samples) into 90-10% stratified train-test sets, denoted as *raw_train_set* and *raw_test_set*. Then, these subsets are processed as follows:

1. *Data preprocessing:* The first step of any machine learning-based classification is to process the raw datasets to fix any missing value. Since each event we monitor returns a numerical value, we replace the missing values, if any, with the arithmetic mean of the respective event.

2. *Feature engineering:* In this step, we obtain features that can be used to train a machine learning model for our prediction problem. Here, we compute 12 statistical functions (listed in Table 6.2) for every event. This step converts each sample consisting of 300 readings (rows) x 28 *events* (columns) to a single row of 336 (28 *events* x 12 features) data-points. The features extracted in this phase, hereinafter referred to as *train_set* and *test_set*, are used for the subsequent stages.

0.2, 0.4, 0.6, and 0.8 quantile	1, 2, and 3 sigma
Arithmetic and geometric mean	Kurtosis
Skewness	Variance

Table 6.2: The statistical functions used for our feature engineering phase

3. *Feature scaling:* It is an essential step to eliminate the influence of large-valued features because features with larger magnitude can dominate the objective function, and thus, deterring an estimator to learn from other features correctly. Hence, we standardize features using standard scaler, which removes the mean and scale the features to unit variance.

4. *Feature selection:* In machine learning, feature selection or dimensionality reduction is the process of selecting a subset of relevant features that are used in model construction. It aims to improve estimators' accuracy as well as to boost their performance on high-dimensional datasets. To do so, we calculate the importance of features using *forests of trees* [32] and select the most relevant features.

5. *Training:* The training phase consists of learning the model parameters for the given classifier on the training set, i.e., *train_set*. Given the nature of the problem, we resort to supervised machine learning procedures. In particular, we employed two of the most successful machine learning methods for classification, namely Random Forest (RF) [116] and Support Vector Machine (SVM) [63].

For model selection, we use grid search with 5-fold Cross Validation (CV). The validated hyper-parameters for RF and SVM are shown in Table 6.3 and Table 6.4, respectively. We chose standard range of values for the hyper-parameters [117].

Parameter	Validated values	Effect on the model
n_ estimators	{10, 25, 50, 100, 125, 150}	Number of trees use in the ensemble.
max_ depth	[2, ∞)	Maximum depth of the trees.
max_ features	'auto', 'log2'	Number of features to consider when looking for the best split.
split_ criterion	*gini, entropy*	Criterion used to split a node in a decision tree.
bootstrap	*true, false*	Bootstrap Aggregation (a.k.a. bagging) is a technique that reduces model variances (overfitting) and improves the outcome of learning on limited sample or unstable datasets.
random_ state	10	The seed used by the random number generator.

Table 6.3: Hyper-parameters validated for RF classifier

Parameter	Validated values	Effect on the model
kernel	'rbf', 'poly', 'sigmoid'	Specifies the kernel type to be used in the algorithm.
C	[10^{-3}, 10^{5}]	Regularization parameter that controls the trade-off between the achieving a low training error and a low testing error that is the ability to generalize your classifier to unseen data.
γ	'auto', [10^{-7}, 10^{3}]	Shape parameter of the RBF kernel which defines how an example influence in the final classification.
degree	default=3	Degree of the polynomial kernel function ('poly'). Ignored by all other kernels.
random_ state	10	The seed of the pseudo random number generator used when shuffling the data for probability estimates.

Table 6.4: Hyper-parameters validated for SVM classifier

6. *Prediction:* Finally, prediction is made on *test_ set*.

The process is repeated ten times for a given experiment and the final results are computed over these ten runs.

6.2 Evaluation

We throughly evaluated our approach by performing an exhaustive set of experiments. We performed the following six different experiments: (1) *binary* classification; (2) *currency* classification; (3) *nested* classification; (4) *sample*

length; (5) *feature relevance*; and (6) *unseen-miner program* classification. Table 6.5 describes the sample distribution in our base-dataset for each system, i.e., *S1* and *S2*. Here, sub-classes of the mining task refer to the cryptocurrencies (discussed in Section 6.1.3) while sub-classes of the non-mining task refer to the actual user-tasks that belong to the negative class (mentioned in Section 6.1.2). We use the entire base-dataset (1100 samples per system) for each experiment, unless otherwise stated in an experiment.

Task	**Sub-classes per task**	**Samples per sub-class**	**Total samples per task**
Mining	11	50	550
Non-mining	11	50	550

Table 6.5: Dataset: name of the task, sub-classes per task, samples per sub-class, and total samples per task for each system

We evaluated our classifier using standard classification metrics: Accuracy, Precision, Recall, and F_1 score. To increase the statistical significance of our results, we report the mean and the margin of error for the results with 95% confidence interval from ten runs of each experiment for each of the evaluation metric. See Appendix B for details of these evaluation metrics and the related statistical terms. We use $(\cdot)$ indicates the best result for the metric and report the results as $mean \pm margin\ of\ error$.

6.2.1 Binary Classification

Our main goal is to identify whether a given instance represents the mining task or not. Hence, in this experiment, the label of each sample was defined as the positive or negative class, accordingly. Table 6.6 presents the results of the *binary* classification using both RF and SVM.

System	**Method**	**Accuracy**	**Precision**	**Recall**	**F1**
S1	RF	$1.000 \pm 0.000\cdot$	$1.000 \pm 0.000\cdot$	$1.000 \pm 0.000\cdot$	$1.000 \pm 0.000\cdot$
	SVM	0.999 ± 0.002	0.999 ± 0.002	0.999 ± 0.002	0.999 ± 0.002
S2	RF	$0.999 \pm 0.002\cdot$	$0.999 \pm 0.002\cdot$	$0.999 \pm 0.002\cdot$	$0.999 \pm 0.002\cdot$
	SVM	0.990 ± 0.018	0.991 ± 0.016	0.990 ± 0.018	0.990 ± 0.018

Table 6.6: Results for *binary* classification

Both the RF and SVM yielded superior performance. However, RF performed better than SVM on both the system, i.e., *S1* and *S2*. For the same reason, we report the results only for RF for the subsequent experiments.

6.2.2 Currency Classification

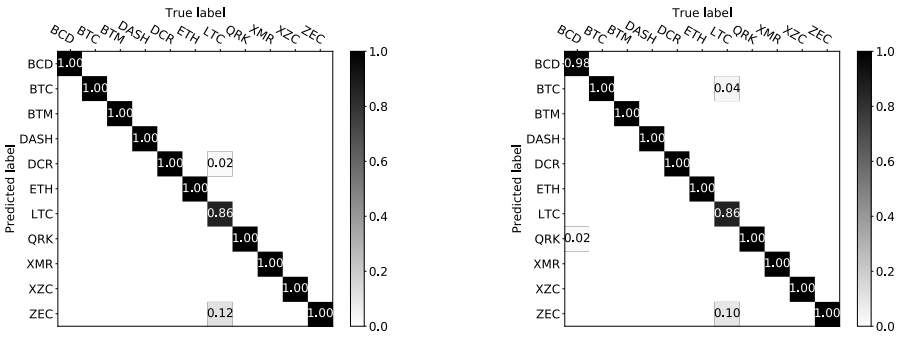
The aim of this experiment is to understand the difficulty level of classification among various cryptocurrencies. Therefore, the input dataset for this

experiment contained instances only of the cryptocurrencies. Table 6.7 lists the results of the *currency* classification.

System	Accuracy	Precision	Recall	F1
S1	0.987 ± 0.017	0.992 ± 0.011	0.988 ± 0.016	0.985 ± 0.020
S2	0.986 ± 0.018	0.981 ± 0.027	0.985 ± 0.018	0.982 ± 0.024

Table 6.7: Results for *currency* classification

Figure 6.2 depicts the confusion matrices for the classification among various cryptocurrencies to provide a better perception of the results. Here, Figure 6.2 (a) and Figure 6.2 (b) correspond to *S1* and *S2*, respectively. The confusion matrices are drawn using the aggregate results from all the ten runs. *Currency* classification is a multi-class classification problem, and some cryptocurrencies were misclassified among each other (see Figure 6.2). Hence, the results are slightly lower than that of the *binary* classification.



(a) *S1* (b) *S2*

Figure 6.2: Confusion matrix for classification among various currencies

6.2.3 Nested Classification

This experiment represents a simulation of a real-world scenario. Here, we first classify whether a given instance belongs to the positive class. If so, we identify the cryptocurrency it belongs to. Essentially, *nested* classification is equivalent to performing *currency* classification on the instances classified as positive in the *binary* classification. Figure 6.3 depicts the hierarchy of *nested* classification.

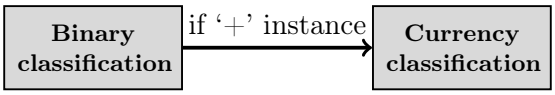


Figure 6.3: Hierarchy of *nested* classification

Table 6.8 shows the results of the *nested* classification. In the worst case, we expect the outcome of this experiment to be lower than that of the *binary* classification and *currency* classification together because a crucial aspect of such staged classification is that an error made in the prediction during the primary stage influences the subsequent stage; the results for *S1* shows this phenomenon. However, in a common scenario, the expected outcome of this experiment would be between the results for the *binary* classification and *currency* classification; the results for *S2* shows this effect.

System	Accuracy	Precision	Recall	F1
S1	0.973 ± 0.020	0.972 ± 0.026	0.972 ± 0.020	0.967 ± 0.026
S2	0.996 ± 0.007	0.997 ± 0.006	0.996 ± 0.008	0.996 ± 0.008

Table 6.8: Results for *nested* classification

6.2.4 Sample Length

The objective of this experiment is to understand the effect of length of the samples. For deployment in the real-world scenario, any solution - apart from being accurate - must be able to detect cryptojackers rapidly. To this end, we performed the *binary* classification of samples of a length of 5, 10, 15, 20, 25, and 30 seconds, each in separate experiments. It is worth mentioning that we used samples of identical length for both the training and testing. Figure 6.4 shows the F_1 score when using samples of different length.

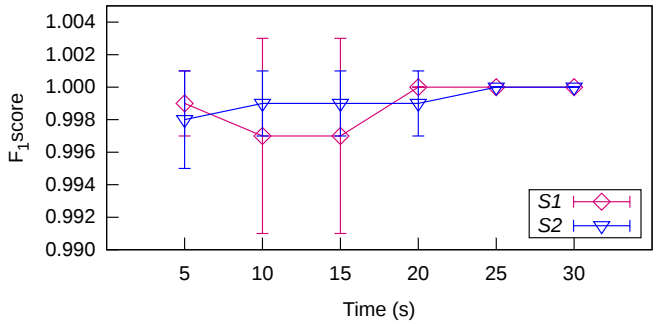


Figure 6.4: F_1 score when using samples of different length (whiskers represent margin of error)

As explained in Section 6.1.1, the task of mining is to repeatedly execute the core PoW algorithm. Hence, even samples of shorter length can grasp the signature. As shown in Figure 6.4, our system can achieve high performance with samples of 5 seconds. The dip in the curve for *S1* corresponds to the thousandths digit of the F_1 score. For the sake of brevity, we omitted the results for sample shorter than five seconds and only focus on the required minimum sample length to attain high performance with our solution.

6.2.5 Feature Relevance

Next, we focus on our feature selection process (see Section 6.1.4). After calculating the importance of features, we sorted them in the ascending order of their importance and selected the first-Ψ% features to do the *binary* classification. Figure 6.5 depicts the F_1 score when using first-Ψ% features.

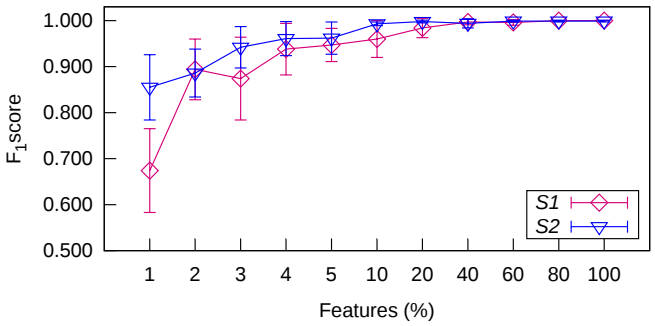


Figure 6.5: F_1 score when using first-Ψ% features (whiskers represent margin of error)

Since the features are sorted in the ascending order of their importance, we begin with the feature with lowest significance. Intuitively, including important features further improves the classification process. As shown in the Figure 6.5, our classifier attains high performance on both the systems using only the first-40% (less relevant) features, which verifies/approves our feature engineering and selection process.

6.2.6 Unseen-miner Program Classification

There can be several different miner-programs to mine a given cryptocurrency. These programs come from different sources. Consequently, there can be some variations in the behavior of the miner-program itself, e.g., in the code section before/after the PoW function or handling (on the programming-side) a correct nonce found while mining. The reason is that they are developed by different developers, which intuitively causes variations. Training a model for each program may not be feasible for a variety of reasons. Hence, to investigate the effectiveness of our approach, we set up another experiment. Here, we selected the *binary* classification as the target. However, we

chose two additional miner programs for BTC, namely, BFGMiner 5.5 and cgminer 4.10. We collected additional 50 samples each for BFGMiner 5.5 and cgminer 4.10 on both S1 and S2 separately. In the training phase, we used samples from one of the three miner programs for BTC. On the contrary, we used samples from one of the other two miner programs for BTC during the testing phase. Table 6.9 presents the results of classifying samples from the miner programs that were unseen in the training phase.

System	Task	Accuracy	Precision	Recall	F1
S1	α_β	0.997 ± 0.006	0.997 ± 0.006	0.997 ± 0.006	0.997 ± 0.006
	α_γ	0.998 ± 0.005	1.000 ± 0.000	0.997 ± 0.006	0.998 ± 0.004
	β_α	1.000 ± 0.000	1.000 ± 0.000	1.000 ± 0.000	1.000 ± 0.000
	β_γ	0.999 ± 0.001	0.999 ± 0.002	0.999 ± 0.002	0.999 ± 0.002
	γ_α	0.999 ± 0.002	0.999 ± 0.002	0.999 ± 0.002	0.999 ± 0.002
	γ_β	1.000 ± 0.000	1.000 ± 0.000	1.000 ± 0.000	1.000 ± 0.000
S2	α_β	0.999 ± 0.001	0.999 ± 0.002	0.999 ± 0.002	0.999 ± 0.002
	α_γ	0.998 ± 0.002	0.997 ± 0.003	0.997 ± 0.003	0.997 ± 0.003
	β_α	0.999 ± 0.002	0.998 ± 0.003	0.998 ± 0.003	0.998 ± 0.003
	β_γ	0.999 ± 0.001	0.999 ± 0.002	0.999 ± 0.002	0.999 ± 0.002
	γ_α	0.999 ± 0.001	0.999 ± 0.002	0.999 ± 0.002	0.999 ± 0.002
	γ_β	0.999 ± 0.001	0.999 ± 0.002	0.999 ± 0.002	0.999 ± 0.002

Table 6.9: Results for *unseen-miner program* classification

The notation X_Y means that the training was done with the samples from X while the testing was done with the sample from Y for BTC. Here, α = cpuminer-multi 1.3.4, β = BFGMiner 5.5, and γ = cgminer 4.10. It is important to mention that these results are for the classification of all the mining and non-mining tasks with BTC being trained and tested upon samples from different programs.

As discussed in Section 6.1.1, the miners have to execute the same core PoW algorithm for a given cryptocurrency. Hence, samples from different miner programs for the same cryptocurrency retain the same signatures, which is reflected in our results.

6.3 Discussion

In this section, we address the potential limitation of our proposed approach.

6.3.1 Zero-day Attack

As explained in Section 5.5, a zero-day cryptocurrency would be a currency that uses a completely new/custom and unseen PoW algorithm. For a cryptocurrency to obtain market value, its core-network should be supported by miners/pools as well as its PoW algorithm must be accepted by the crypto-community and tested mathematically for its robustness. Therefore, the

PoW algorithm for a new cryptocurrency would become public by the time it gets ready for mining, which would give us sufficient time to capture this new cryptocurrency's signature and to train our model. In our experiments, we considered all the popular cryptocurrencies, and our results (presented in Section 6.2) demonstrate the high quality of our proposed approach along various dimensions.

6.3.2 Process Selection

As mentioned in Section 6.1.2, our system requires per program/process-based recording of HPC for different *events* as the input to the classifier. In practice, several processes run in the system. Hence, monitoring each process may consume time and can be seen as a limitation of our work. However, as shown in Figure 6.4, our system can achieve high performance even with samples of 5 seconds. On another side, the miner programs attempt to use all the available resources. Thus, an initial sorting of processes based on their resource usage can help to boost the detection process in real-time.

6.3.3 Scalability

The key concept of our approach is to profile the behavior of a processor's *events* for mining algorithms. Since there are only a finite number of CPUs/GPUs, procuring their signature is only a matter of data collection. However, it might appear as a ponderous job and may be seen as a limitation of our work. But, once it is accomplished for the available CPUs/GPUs, maintaining it is relatively simpler as merely a limited number of CPUs/GPUs are released over a period of time.

6.3.4 Restricted Mining

A mining strategy to evade detection from our proposed methodology can be *restricted mining* that aims to change the footprint of the mining process. Here, the miner program/process can be modified to perform arbitrary operations during mining. But, such maneuvers would directly affect the hashing rate and consequently the profits; making the task of mining less appealing. Nevertheless, like any signature-based detection technique, it may be seen as a limitation of our work.

6.4 Summary

Cybercriminals have developed several proficient ways to exploit cryptocurrencies with an aim to commit many unconventional financial frauds. Cryptojacking is one of the most recent means to monetize the computational power of the victims. In this chapter, we present our efficient methodology

to identify cryptojacking. Our solution has a broader scope as it targets the core PoW algorithms and uses the low-performance overhead HPC that are present in modern processors to create discernible signatures. We tested our generic approach against a set of rigorous experiments that include eleven distinct cryptocurrencies. We found that our classifier attains high performance even with short samples of five seconds.

Chapter 7

Conclusions

The popularity of cryptocurrencies has increased enormously over the last decade. Many cryptocurrencies, e.g., Bitcoin, are now recognized as a regular mode of payment. With a growing demand for cryptocurrencies, new currencies are introduced each day. At the time of writing, there are over 2000 active cryptocurrencies in the market. Typically, these virtual currency systems aim to provide pseudo-anonymity to hide the real identity of the payer and payee. The pseudo-anonymity offered by these cryptocurrencies has, unfortunately, made such currencies a handy utility among cybercriminals. Apart from giving birth to novel cybercrimes, these currencies also come with novel security and privacy concerns.

In this thesis, we focused on the recently thriving, prominent, and severe concerns arising due to cryptocurrencies. In particular, we focused on issues related to Bitcoin in Part I, the security flaw in Algorand's truly democratic blockchain consensus protocol in Part II, and covert cryptomining associated with various cryptocurrencies in Part III. We first summarize our contributions to the state-of-the-art in Section 7.1 and outline the possible future works in Section 7.2.

7.1 Summary of Contributions

In this section, we summarize the contributions of our works presented in this thesis.

7.1.1 Bitcoin

In Part I, we investigated issues related to the most popular cryptocurrency, i.e., Bitcoin. Out of several concerns associated with Bitcoin, we studied two issues that hold significant importance in this era of cryptocurrencies. In particular, we focused on rapidly increasing ransomware campaigns and

the privacy concerns related to the Bitcoin wallet apps. For the former, we presented our comprehensive and longitudinal study on the recent ransomware attacks and reported the economic impact of such ransomware from the Bitcoin payment perspective. For the latter, we presented our study on identifying sensitive user activities on smartphone-based Bitcoin wallet apps that are commonly used for sending, receiving, and trading Bitcoin.

- *Bitcoin Ransomware Campaigns:* To the best of our knowledge, our work presented in Chapter 2 is the first study that discusses the characteristics and functionality of various Bitcoin ransomware, as well as it also gives more accurate insights on the economic impact of such ransomware. To this end, we presented a lightweight framework that can identify, collect, and analyze Bitcoin addresses that belong to the same user(s). Next, we proposed our novel approach to classify a payment as ransom. We used our framework to analyze the economic impact (in terms of ransoms extorted in Bitcoin) of those ransomware: (i) that used Bitcoin as at least one mode of ransom payment and (ii) for which at least one Bitcoin address was publicly known.

- *Privacy Issues in Bitcoin Wallet Apps:* In Chapter 3, we aimed to identify a set of sensitive and fundamental user actions on smartphone-based Bitcoin wallet apps. To do so, we resorted to analyzing encrypted network traffic that elicits from these apps in response to the user's interaction. We built a complete implementation of a machine learning-based framework to perform the analysis. We designed our experiments to reasonably simulate the real-world scenarios, where traffic traces from both Android and iOS devices were classified in nested classification layers. Our framework can help to scrutinize potential Bitcoin wallet users, which consequently, may help in the hunt of cybercriminals that exploit Bitcoin in various forms.

7.1.2 Algorand

In Part II, we investigated Algorand that is a recently proposed innovative blockchain consensus protocol. To the best of our knowledge, it is the first formal study on Algorand. In particular, we presented a security analysis of Algorand. Algorand guarantees an overwhelming probability of linearity of the blockchain. Moreover, it is designed to solve the "blockchain trilemma" of decentralization, scalability, and security. Hence, our aim here was to analyze and further strengthen such a truly democratic consensus protocol.

- *Security Flaw in Truly Democratic Consensus Protocol:* In Chapter 4, we demonstrated a practically feasible attack on the Algorand protocol. We evaluated the impact of our proposed attack in our simulator that we created from scratch in Java. We found that an adversary using our

attack can target a group of honest users and slow down the consensus process, which consequently, leaves the targeted nodes behind in the chain. Moreover, we elaborated the major factors that make such an attack scenario challenging for honest nodes. Finally, we also presented possible countermeasures to prevent such an attack.

7.1.3 Cryptominers

With the increase in the popularity of cryptocurrencies, the demand for cryptomining has increased drastically. In parallel to legitimate mining demands, covert cryptomining has emerged as a utility for malicious actors to gain financial incentives. Cryptocurrencies, such as Monero, has enabled even naive users to mine via a browser application. We have investigated covert cryptomining in Part III and proposed two efficient solutions to detect it under different scenarios.

- *Detecting Covert Miners via Magnetic Side-channel:* In Chapter 5, we considered both conscious-miners and unconscious-miners. Here, we leveraged the magnetic side-channel to detect covert cryptomining. We built a complete implementation of our system using advanced machine learning techniques. We considered twelve different cryptocurrencies in our experiments, which indeed are the most mined cryptocurrencies. We designed and performed five different experiments to thoroughly assess the quality of our approach. Our proposed unique approach works even when the investigator does not have root-privileges or login-access on the suspect device.
- *Detecting Covert Miners via Hardware Performance Counters:* In Chapter 6, we specifically focused on unconscious-mining. However, unlike the previous case, the investigator here owns the device and can elevate his/her privileges to the root-level. We utilized HPC to create clean signatures of the execution patterns of the underlying PoW algorithms on a given processor. In a battery of assessments of our proposed generic approach, we found that our approach can near-perfectly detect cryptominers for various cryptocurrencies.

7.2 Future Work

In this section, we present the possible future directions of our research contributions reported in this thesis.

7.2.1 Bitcoin

Despite being one of the most comprehensive and accurate studies on ransomware campaigns, there is still some room for significant improvements in our

work discussed in Chapter 2. In particular, we hope to improve our work to deal with its limitations, i.e., overestimation and underestimation. Moreover, we will extend our framework to other cryptocurrencies. We will also investigate the ransoms extorted via other payment options; we hope to present a comprehensive report that will include ransom payments from all payment option endorsed by the ransomware. Finally, we will attempt to trace how the received ransoms were used and by whom. As a future direction of our proposal in Chapter 3, we will investigate the security and privacy implication of transacting on such apps by considering a stronger adversary model. On the other hand, we also intend to find efficient countermeasures to preserve the privacy of the Bitcoin wallet app users. Finally, we will also explore the possibility to deanonymize a financial transaction placed via the wallet apps.

7.2.2 Algorand

As far as the work presented in Chapter 4 is concerned, we will evaluate our proposed attack on the actual test-network of Algorand's real-world implementation. The rigorous formalization of our proposed countermeasures is also kept as one of the future efforts related to this work. We will also work in the direction to find even more effective countermeasures with an aim to make Algorand more secure and reliable.

7.2.3 Cryptominers

Extending our work described in Chapter 5, we will investigate the variations in the magnetic profiles of even more processors. We will explore the possibility of creating a common profile across different processors for a given PoW algorithm. We will also evaluate the performance of our approach under different mining rate as well as in scenarios with varying magnetic field profiles, e.g., server rooms. Finally, we hope to release a smartphone app for real-time identification of covert cryptomining. For our work introduced in Chapter 6, we will investigate the variations in the samples from different operating system and virtualized environments. We also hope to release a desktop application for real-time identification of covert cryptomining.

We will also work in the direction to: (1) preserve privacy and improve security for the legitimate cryptocurrency users; (2) make cryptocurrency systems more democratic by enhancing their fundamental protocols; and (3) prevent exploitations of pseudo-anonymity offered by cryptocurrencies.

Bibliography

[1] Coinhive. [online] https://tinyurl.com/ybsy89k2.

[2] CoinMarketCap. [online] https://tinyurl.com/o94fhlw.

[3] Crypto-Loot. [online] https://tinyurl.com/y76ppd5g.

[4] The *nvprof* Tool. [online] https://tinyurl.com/y8tqxn74.

[5] The *perf* Tool. [online] https://tinyurl.com/ybpmxw8.

[6] The *stress-ng* Tool. [online] https://tinyurl.com/my6ehnj.

[7] Enigma: A Private, Secure and Untraceable Transactions System for Cloackcoin. [online] https://tinyurl.com/yywlw4rg, 2014.

[8] Nxt Whitepaper. [online] https://tinyurl.com/y5sujywz, 2014.

[9] US Government Bans Researcher for Supercomputer Bitcoin Mining. [online] https://tinyurl.com/y9w8qq59, 2014.

[10] Hyperledger Sawtooth Documentation. [online] https://tinyurl.com/yxnzu7qt, 2016.

[11] An Italian Bank's Server was Hijacked to Mine Bitcoin. [online] https://tinyurl.com/yac8c8jq, 2017.

[12] CryptoJacking Android Malware 'Loapi' Can Physically Damage Your Device. [online] https://tinyurl.com/yazv85ay, 2017.

[13] NYC Government Employee Caught Mining Bitcoin On Work Computer. [online] https://tinyurl.com/y7vb3nlj, 2017.

[14] Persistent Drive-by Cryptomining Coming to a Browser Near You. [online] https://tinyurl.com/yd5roadb, 2017.

[15] Smoothing of a 1D signal. [online] https://tinyurl.com/ya9z9vgq, 2017.

[16] Banking Is Only The Beginning: 42 Big Industries Blockchain Could Transform. [online] https://tinyurl.com/yawbz6tt, 2018.

[17] Bitcoin Mining Pools. [online] https://tinyurl.com/y8pdk922, 2018.

[18] College Students Use Free Electricity on Campus to Mine Bitcoin. [online] https://tinyurl.com/ybue72el, 2018.

[19] Cryptojacking Attack Found on Los Angeles Times Website. [online] https://tinyurl.com/y8ghcvmd, 2018.

[20] FacexWorm Targets Cryptocurrency Trading Platforms, Abuses Facebook Messenger for Propagation. [online] https://tinyurl.com/yd2zja9q, 2018.

[21] Greedy Cybercriminals Host Malware on GitHub. [online] https://tinyurl.com/y9qon8ch, 2018.

[22] Is Bitcoin Mining Profitable or Worth it in 2018? [online] https://tinyurl.com/ybnydb8g, 2018.

[23] Nuclear Engineers Arrested for Mining Cryptocurrency Using Government Supercomputer. [online] https://tinyurl.com/y9kdf5ts, 2018.

[24] OpenStack. [online] https://tinyurl.com/gv8jsjc, 2018.

[25] Revenues Down, Hashrates Up: 2018 Mining Outlook By The Numbers. [online] https://tinyurl.com/yc586s9v, 2018.

[26] Rogue Employees Mine Cryptocurrency Using Company Hardware. [online] https://tinyurl.com/y8u6s524, 2018.

[27] rTorrent Client Exploited in the Wild to Deploy Monero Cryptominer. [online] https://tinyurl.com/yaqy7u3k, 2018.

[28] Tesla Hackers Hijacked Amazon Cloud Account to Mine Cryptocurrency. [online] https://tinyurl.com/y9epv3do, 2018.

[29] UK ICO, USCourts.gov... Thousands of Websites Hijacked by Hidden Cryptomining Code after Popular Plug-in Pwned. [online] https://tinyurl.com/y7upaxgv, 2018.

[30] Ukrainian Professor Mines Bitcoin at School. [online] https://tinyurl.com/ybglcusy, 2018.

[31] WebCobra Malware Uses Victims' Computers to Mine Cryptocurrency. [online] https://tinyurl.com/ycuhowb3, 2018.

[32] Feature Importances with Forests of Trees. [online] https://tinyurl.com/y3nlad2h, 2019.

[33] Under the Hood of Cyber Crime. [online] https://tinyurl.com/ydhauj8x, 2019.

[34] F. Aiolli, M. Conti, A. Gangwal, and M. Polato. Mind Your Wallet's Privacy: Identifying Bitcoin Wallet Apps and User's Actions through Network Traffic Analysis. In ACM Symposium on Applied Computing (SAC), pages 1484–1491, 2019.

[35] G. Al-Naymat, S. Chawla, and J. Taheri. SparseDTW: A Novel Approach to Speed Up Dynamic Time Warping. In Australasian Data Mining Conference (AusDM), pages 117–127, 2009.

[36] H. F. Alan and J. Kaur. Can Android Applications be Identified using only TCP/IP Headers of their Launch Time Traffic? In ACM conference on Security and Privacy in Wireless and Mobile Networks (WiSec), pages 61–66, 2016.

[37] G. Ateniese, B. Hitaj, L. V. Mancini, N. V. Verde, and A. Villani. No Place to Hide that Bytes won't Reveal: Sniffing Location-Based Encrypted Traffic to Track a User's Position. In Springer Network and System Security (NSS), volume 9408, pages 46–59, 2015.

[38] Avast Threat Intelligence Team. Inside Petya and Mischa ransomware. [online] https://tinyurl.com/yyacsmm6, 2016.

[39] A. Bagnall, J. Lines, A. Bostrom, J. Large, and E. Keogh. The Great Time Series Classification Bake Off: A Review and Experimental Evaluation of Recent Algorithmic Advances. Springer Data Mining and Knowledge Discovery, 31(3):606–660, 2017.

[40] I. Bentov, R. Pass, and E. Shi. Snow White: Provably Secure Proofs-of-Stake. IACR Cryptology ePrint Archive, 2016:919, 2016.

[41] D. J. Berndt and J. Clifford. Using Dynamic Time Warping to Find Patterns in Time Series. In AAAI Workshop on Knowledge Discovery in Databases, pages 359–370, 1994.

[42] S. Biedermann, S. Katzenbeisser, and J. Szefer. Hard Drive Side-channel Attacks using Smartphone Magnetic Field Sensors. In Springer Financial Cryptography and Data Security (FC), pages 489–496, 2015.

[43] A. Biryukov, D. Khovratovich, and I. Pustogarov. Deanonymisation of Clients in Bitcoin P2P Network. In ACM conference on Computer and Communications Security (CCS), pages 15–29, 2014.

[44] S. Bistarelli and F. Santini. Go with the -Bitcoin- Flow, with Visual Analytics. In ACM conference on Availability, Reliability and Security (ARES), pages 1–6, 2017.

[45] Bitcoin Wallet App Installs Surpass 25 Million Since 2014, November Downloads Up 800% Year-over-Year. [online] https://tinyurl.com/y2ocxy9l, 2017.

[46] J. Bonneau, A. Miller, J. Clark, A. Narayanan, J. A. Kroll, and E. W. Felten. SoK: Research Perspectives and Challenges for Bitcoin and Cryptocurrencies. In IEEE Symposium on Security and Privacy (S&P), pages 104–121, 2015.

[47] Broadanalysis Threat Intelligence and Malware Research. Neutrino EK from 104.238.185.187 sends DMA Locker 4.0. [online] https://tinyurl.com/y3ujlns9, 2016.

[48] V. Buterin. Slasher: A Punitive Proof-of-Stake Algorithm. [online] https://tinyurl.com/y2yg38lz, 2015.

[49] V. Buterin and V. Griffith. Casper the Friendly Finality Gadget. arXiv preprint: 1710.09437, 2017.

[50] C. Xiao and J. Chen. New OS X Ransomware KeRanger Infected Transmission BitTorrent Client Installer. [online] https://tinyurl.com/zybzhqz, 2016.

[51] X. Cai, X. Zhang, B. Joshi, and R. Johnson. Touching From a Distance: Website Fingerprinting Attacks and Defenses. In ACM Computer and Communications Security (CCS), pages 605–616, 2012.

[52] D. Chaum. Blind Signatures for Untraceable Payments. In Advances in Cryptology, pages 199–203, 1983.

[53] N. V. Chawla, K. W. Bowyer, L. O. Hall, and W. P. Kegelmeyer. SMOTE: Synthetic Minority Over-sampling Technique. Journal of Artificial Intelligence Research, 16(1):321–357, 2002.

[54] J. Chen and S. Micali. Algorand. arXiv preprint: 1607.01341, 2016.

[55] M. Chiappetta, E. Savas, and C. Yilmaz. Real-time Detection of Cache-based Side-channel Attacks using Hardware Performance Counters. Elsevier Applied Soft Computing, 49:1162–1174, 2016.

[56] N. Christin. Traveling the Silk Road: A measurement Analysis of a Large Anonymous Online Marketplace. In ACM conference on World Wide Web (WWW), pages 213–224, 2013.

[57] Comodo Cybersecurity. Global Threat Report Q2 2018 Edition. [online] https://tinyurl.com/y8eos9pl, 2018.

[58] M. Conti, A. Gangwal, G. Lain, and S. G. Piazzetta. Detecting Covert Cryptomining using Hardware Performance Counters. Under review at IEEE Transactions on Dependable & Secure Computing, 2019.

[59] M. Conti, A. Gangwal, and S. Ruj. On the Economic Significance of Ransomware Campaigns: A Bitcoin Transactions Perspective. Elsevier Computers & Security, 79:162–189, 2018.

[60] M. Conti, A. Gangwal, and M. Todero. Blockchain Trilemma Solver Algorand has Dilemma over Undecidable Messages. In International Conference on Availability, Reliability and Security (ARES), pages 1–8, 2019.

[61] M. Conti, S. Kumar, C. Lal, and S. Ruj. A Survey on Security and Privacy Issues of Bitcoin. IEEE Communications Surveys & Tutorials, 20(4):3416–3452, 2018.

[62] M. Conti, L. Mancini, R. Spolaor, and N. Verde. Analyzing Android Encrypted Network Traffic to Identify User Actions. IEEE Transactions on Information Forensics & Security, 11:114–125, 2016.

[63] C. Cortes and V. Vapnik. Support Vector Networks. Machine Learning, 20(3):273–297, 1995.

[64] S. E. Coull and K. P. Dyer. Traffic Analysis of Encrypted Messaging Services: Apple iMessage and Beyond. ACM SIGCOMM Computer Communication Review, 44(5):5–11, 2014.

[65] Counter Threat Unit Research Team. WCry Ransomware Analysis. [online] https://tinyurl.com/y5pn3c6r, 2017.

[66] K. Croman, C. Decker, I. Eyal, A. E. Gencer, A. Juels, A. Kosba, A. Miller, P. Saxena, E. Shi, E. G. Sirer, et al. On Scaling Decentralized Blockchains. In Springer Financial Cryptography and Data Security (FC), pages 106–125, 2016.

[67] CryptoLocker Address 1. [online] https://tinyurl.com/oztjd3e.

[68] CryptoLocker Address 2 [online] https://tinyurl.com/yxpob6s7.

[69] CryptoWall Address 1. [online] https://tinyurl.com/y3qubrqa.

[70] CryptoWall Address 10. [online] https://tinyurl.com/gmf5fa2.

[71] CryptoWall Address 11. [online] https://tinyurl.com/yxou2plv.

[72] CryptoWall Address 12. [online] https://tinyurl.com/y4oa4hoz.

[73] CryptoWall Address 2. [online] https://tinyurl.com/pdbw9r5.

[74] CryptoWall Address 3. [online] https://tinyurl.com/y26lujxn.

[75] CryptoWall Address 4. [online] https://tinyurl.com/y67rwwaj.

[76] CryptoWall Address 5. [online] https://tinyurl.com/y29vf6gm.

[77] CryptoWall Address 6. [online] https://tinyurl.com/y4ub2kmc.

[78] CryptoWall Address 7. [online] https://tinyurl.com/yy77rms9.

[79] CryptoWall Address 8. [online] https://tinyurl.com/y5b4ecnw.

[80] CryptoWall Address 9. [online] https://tinyurl.com/yysxvvgq.

[81] Cyber Threat Alliance (CTA). The Illicit Cryptocurrency Mining Threat Report. [online] https://tinyurl.com/yco7cykl, 2018.

[82] S. Dai, A. Tongaonkar, X. Wang, A. Nucci, and D. Song. NetworkProfiler: Towards Automatic Fingerprinting of Android Apps. In IEEE International Conference on Computer Communications (INFOCOM), pages 809–817, 2013.

[83] W. Dai. b-money. [online] https://tinyurl.com/kqz6rgn, 1998.

[84] Dell SecureWorks Counter Threat Unit Threat Intelligence. CryptoWall Ransomware Threat Analysis. [online] https://tinyurl.com/y2wv6nlz, 2014.

[85] J. Demme, M. Maycock, J. Schmitz, A. Tang, A. Waksman, S. Sethumadhavan, and S. Stolfo. On the Feasibility of Online Malware Detection with Performance Counters. In International Symposium on Computer Architecture (ISCA), pages 559–570, 2013.

[86] G. Di Battista, V. Di Donato, M. Patrignani, M. Pizzonia, V. Roselli, and R. Tamassia. BitConeView: Visualization of Flows in the Bitcoin Transaction Graph. In IEEE Symposium on Visualization for Cyber Security (VizSec), pages 1–8, 2015.

[87] R. Dingledine, N. Mathewson, and P. Syverson. Tor: The Second-generation Onion Router. In USENIX Security, pages 1–17, 2004.

[88] C. Dion-Schwarz, D. Manheim, and P. B. Johnston. Terrorist Use of Cryptocurrencies: Technical and Organizational Barriers and Future Threats. Rand Corporation, 2019.

[89] DMA Locker Address 1. [online] https://tinyurl.com/y2bk5jmm.

[90] DMA Locker Address 2. [online] https://tinyurl.com/y5qe6fmd.

[91] DMA Locker Address 3. [online] https://tinyurl.com/y59ekgv6.

[92] DMA Locker Address 4. [online] https://tinyurl.com/y5vcpeex.

[93] DMA Locker Address 5. [online] https://tinyurl.com/y54otawt.

[94] DMA Locker Address 6. [online] https://tinyurl.com/yyguqhvc.

[95] DMA Locker Address 7. [online] https://tinyurl.com/y4dhj8og.

[96] Emsisoft Lab. CryptoDefense: The Story of Insecure Ransomware Keys and Self-serving Bloggers. [online] https://tinyurl.com/lpkm6qe, 2014.

[97] W. Enck, P. Gilbert, S. Han, V. Tendulkar, B.-G. Chun, L. P. Cox, J. Jung, P. McDaniel, and A. N. Sheth. TaintDroid: An Information-Flow Tracking System for Realtime Privacy Monitoring on Smartphones. ACM Transactions on Computer Systems, 32(2):1–29, 2014.

[98] V. Eruhimov, V. Martyanov, and E. Tuv. Constructing High Dimensional Feature Space for Time Series Classification. In European Conference on Principles and Practice of Knowledge Discovery in Databases (ECML/PKDD), volume 4702, pages 414–421, 2007.

[99] S. Eskandari, A. Leoutsarakos, T. Mursch, and J. Clark. A First Look at Browser-based Cryptojacking. arXiv preprint: 1803.02887, 2018.

[100] I. Eyal. The Miner's Dilemma. In IEEE Symposium on Security and Privacy (S&P), pages 89–103, 2015.

[101] H. Falaki, D. Lymberopoulos, R. Mahajan, S. Kandula, and D. Estrin. A First Look at Traffic on Smartphones. In ACM SIGCOMM Internet Measurement Conference (IMC), pages 281–287, 2010.

[102] False Alarm: Silk Road Competitor Black Market Reloaded Staying Online. [online] https://tinyurl.com/yyw9sp7v, 2013.

[103] H. Finney. RPOW - Reusable Proof Of Work. [online] https://tinyurl.com/y5e4m5ul, 2004.

[104] A. Gangwal and M. Conti. Cryptomining cannot Change its Spots: Detecting Covert Cryptomining using Magnetic Side-channel. IEEE Transactions on Information Forensics & Security, in press, 2019.

[105] Gartner Says Worldwide Sales of Smartphones Recorded First Ever Decline During the Fourth Quarter of 2017. [online] https://tinyurl.com/y5u5vyn6, 2018.

[106] Y. Gilad, R. Hemo, S. Micali, G. Vlachos, and N. Zeldovich. Algorand: Scaling Byzantine Agreements for Cryptocurrencies. In ACM Symposium on Operating Systems Principles, pages 51–68, 2017.

[107] GoldenEye Address 1. [online] https://tinyurl.com/yyerdytg.

[108] GoldenEye Address 2. [online] https://tinyurl.com/y6dan8sd.

[109] GoldenEye Address 3. [online] https://tinyurl.com/y6gvvggh.

[110] M. Guri, A. Daidakulov, and Y. Elovici. MAGNETO: Covert Channel between Air-gapped Systems and Nearby Smartphones via CPU Generated Magnetic Fields. arXiv preprint: 1802.02317, 2018.

[111] M. Guri, B. Zadov, A. Daidakulov, and Y. Elovici. ODINI: Escaping Sensitive Data from Faraday-caged, Air-gapped Computers via Magnetic Fields. arXiv preprint: 1802.02700, 2018.

[112] S. Haber and W. S. Stornetta. How to Time-stamp a Digital Document. In Conference on the Theory and Application of Cryptography (CRYPTO), pages 437–455, 1990.

[113] H. Ham and M. Choi. Applicaion-level Traffic Analysis of Smartphone Users using Embedded Agents. In Asia-Pacific Network Operations and Management Symposium (APNOMS), pages 1–4, 2012.

[114] D. Herrmann, R. Wendolsky, and H. Federrath. Website Fingerprinting: Attacking Popular Privacy Enhancing Technologies with the Multinomial Naïve-Bayes Classifier. In ACM Cloud Computing Security Workshop (CCSW), pages 31–42, 2009.

[115] A. Hintz. Fingerprinting Websites using Traffic Analysis. In Privacy Enhancing Technologies (PET), volume 2482, pages 171–178, 2003.

[116] T. K. Ho. Random Decision Forests. In International Conference on Document Analysis and Recognition (ICDAR), pages 278–282, 1995.

[117] C.-W. Hsu, C.-C. Chang, and C.-J. Lin. A Practical Guide to Support Vector Classification. Technical report, 2003.

[118] D. Y. Huang, H. Dharmdasani, S. Meiklejohn, V. Dave, C. Grier, D. McCoy, S. Savage, N. Weaver, A. C. Snoeren, and K. Levchenko. Botcoin: Monetizing Stolen Cycles. In Network and Distributed System Security Symposium (NDSS), pages 1–16, 2014.

[119] D. Y. Huang, D. McCoy, M. M. Aliapoulios, V. G. Li, L. Invernizzi, E. Bursztein, K. McRoberts, J. Levin, K. Levchenko, and A. C. Snoeren. Tracking Ransomware End-to-end. In IEEE Symposium on Security and Privacy (S&P), pages 1–14, 2018.

[120] K. Jarvis. CryptoLocker Ransomware. [online] https://tinyurl.com/y2jm8ppx, 2013.

[121] joostbijl. CryptoLocker Ransomware Intelligence Report. [online] https://tinyurl.com/yxksdpon, 2014.

[122] J.P. Morgan Creates Digital Coin for Payments. [online] https://tinyurl.com/y492gah7, 2019.

[123] E. Keogh and C. A. Ratanamahatana. Exact Indexing of Dynamic Time Warping. Springer Knowledge and Information Systems, 7(3):358–386, 2005.

[124] KeRanger Address 1. [online] https://tinyurl.com/y3t97bcu.

[125] KeRanger Address 2. [online] https://tinyurl.com/yxhy2zs3.

[126] KeRanger Address 3. [online] https://tinyurl.com/y4qgu8po.

[127] KeRanger Address 4. [online] https://tinyurl.com/yybmtcqx.

[128] KeRanger Address 5. [online] https://tinyurl.com/y5658cxy.

[129] KeRanger Address 6. [online] https://tinyurl.com/y6aplh3j.

[130] A. Kharraz, W. Robertson, D. Balzarotti, L. Bilge, and E. Kirda. Cutting the Gordian Knot: A Look Under the Hood of Ransomware Attacks. In Springer conference on Detection of Intrusions and Malware, and Vulnerability Assessment (DIMVA), pages 3–24, 2015.

[131] A. Kiayias, A. Russell, B. David, and R. Oliynykov. Ouroboros: A Provably Secure Proof-of-Stake Blockchain Protocol. In International Cryptology Conference (CRYPTO), pages 357–388, 2017.

[132] S. King. Primecoin: Cryptocurrency with Prime Number Proof-of-Work. [online] https://tinyurl.com/oyhaaw8, 2013.

[133] S. King and S. Nadal. PPcoin: Peer-to-Peer Cryptocurrency with Proof-of-Stake. [online] https://tinyurl.com/yxu2phj8, 2012.

[134] J. V. Kistowski, H. Block, J. Beckett, C. Spradling, K.-D. Lange, and S. Kounev. Variations in CPU Power Consumption. In ACM/SPEC International Conference on Performance Engineering (ICPE), pages 147–158, 2016.

[135] R. K. Konoth, E. Vineti, V. Moonsamy, M. Lindorfer, C. Kruegel, H. Bos, and G. Vigna. MineSweeper: An In-depth Look into Drive-by Cryptocurrency Mining and Its Defense. In ACM Conference on Computer and Communications Security (CCS), pages 14–30, 2018.

[136] S.-W. Lee, J.-S. Park, H.-S. Lee, and M.-S. Kim. A Study on Smartphone Traffic Analysis. In IEEE Asia-Pacific Network Operations and Management Symposium (APNOMS), pages 1–7, 2011.

[137] W. Li, S. Andreina, J.-M. Bohli, and G. Karame. Securing Proof-of-Stake Blockchain Protocols. In Data Privacy Management - Cryptocurrencies and Blockchain Technology (DPM-CBT), pages 297–315. 2017.

[138] K. Liao, Z. Zhao, A. Doupé, and G.-J. Ahn. Behind Closed Doors: Measurement and Analysis of CryptoLocker Ransoms in Bitcoin. In APWG eCrime Research, pages 1–13, 2016.

[139] M. Liberatore and B. N. Levine. Inferring the Source of Encrypted HTTP Connections. In ACM Computer and Communications Security (CCS), pages 255–263, 2006.

[140] J. Liu, Z. Zhao, X. Cui, Z. Wang, and Q. Liu. A Novel Approach for Detecting Browser-based Silent Miner. In IEEE conference on Data Science in Cyberspace (DSC), pages 490–497, 2018.

[141] LogRhythm Labs. NotPetya Technical Analysis. `[online] https://tinyurl.com/yxdm2qbr`, 2017.

[142] L. Luu, R. Saha, I. Parameshwaran, P. Saxena, and A. Hobor. On Power Splitting Games in Distributed Computation: The Case of Bitcoin Pooled Mining. In IEEE Computer Security Foundations Symposium (CSF), pages 397–411, 2015.

[143] M. Lee, W. Mercer, P. Rascagneres, and C. Williams. Player 3 Has Entered the Game: Say Hello to 'WannaCry'. `[online] https://tinyurl.com/l6c7yyf`, 2017.

[144] Malwarebytes Labs. DMA Locker 4.0: Known ransomware preparing for a massive distribution. `[online] https://tinyurl.com/y2mw7k7b`, 2016.

[145] E. Mateos and C. H. Gebotys. Side Channel Analysis using Giant Magneto Resistive (GMR) Sensors. In International Workshop on Constructive Side-Channel Analysis and Secure Design (COSADE), pages 42–49, 2011.

[146] N. Matyunin, J. Szefer, S. Biedermann, and S. Katzenbeisser. Covert Channels using Mobile Device's Magnetic Field Sensors. In Asia and South Pacific Design Automation Conference (ASP-DAC), pages 525–532, 2016.

[147] D. McGinn, D. Birch, D. Akroyd, M. Molina-Solana, Y. Guo, and W. J. Knottenbelt. Visualizing Dynamic Bitcoin Transaction Patterns. Big Data, 4(2):109–119, 2016.

[148] G. Medvinsky and C. Neuman. NetCash: A Design for Practical Electronic Currency on the Internet. In ACM conference on Computer and Communications Security (CCS), pages 102–106, 1993.

[149] S. Meiklejohn, M. Pomarole, G. Jordan, K. Levchenko, D. McCoy, G. M. Voelker, and S. Savage. A Fistful of Bitcoins: Characterizing Payments Among Men with No Names. In ACM Internet Measurement Conference (IMC), pages 127–140, 2013.

[150] S. Micali, M. Rabin, and S. Vadhan. Verifiable Random Functions. In IEEE Annual Symposium on Foundations of Computer Science, pages 120–130, 1999.

[151] Mischa Address 1. [online] https://tinyurl.com/yya3pn25.

[152] T. Mitsa. Temporal Data Mining. Chapman and Hall Press, 2010.

[153] S. Mongkolluksamee, V. Visoottiviseth, and K. Fukuda. Combining Communication Patterns & Traffic Patterns to Enhance Mobile Traffic Identification Performance. Journal of Information Processing, 24(2):247–254, 2016.

[154] C. Mora, R. Rollins, K. Taladay, M. Kantar, M. Chock, M. Shimada, and E. C. Franklin. Bitcoin Emissions Alone could Push Global Warming above 2 °C. Nature Climate Change, 8(11):931–933, 2018.

[155] M. Müller, H. Mattes, and F. Kurth. An Efficient Multiscale Approach to Audio Synchronization. In International Conference on Music Information Retrieval (ISMIR), pages 192–197, 2006.

[156] S. Nakamoto. Bitcoin: A Peer-to-Peer Electronic Cash System. [online] https://tinyurl.com/3f4a6lr, 2008.

[157] S. Nakamoto. Bitcoin v0.1 Released. [online] https://tinyurl.com/y4wb2c7y, 2009.

[158] T. T. Nguyen and G. Armitage. A Survey of Techniques for Internet Traffic Classification using Machine Learning. IEEE Communications Surveys & Tutorials, 10(4):56–76, 2008.

[159] NotPetya Address 1. [online] https://tinyurl.com/yyx4sdqt.

[160] Number of Smartphones Sold to End Users Worldwide from 2007 to 2020. [online] https://tinyurl.com/yylypl6q, 2019.

[161] NY School Delays Start of Year After Ransomware Attack. [online] https://tinyurl.com/y3x4zvsh, 2019.

[162] K. J. O'Dwyer and D. Malone. Bitcoin Mining and its Energy Footprint. 2014.

[163] P4Titan. Slimcoin. A Peer-to-Peer Cryptocurrency with Proof-of-Burn. [online] https://tinyurl.com/yx9vonug, 2014.

[164] A. Panchenko, L. Niessen, A. Zinnen, and T. Engel. Website Fingerprinting in Onion Routing Based Anonymization Networks. In ACM Workshop on Privacy in the Electronic Society (WPES), pages 103–114, 2011.

[165] C. Percival. Stronger Key Derivation via Sequential Memory-hard Functions. [online] https://tinyurl.com/oms2uoa, 2009.

[166] D. Pike, P. Nosker, D. Boehm, D. Grisham, and S. Woods. PoST White Paper. [online] https://tinyurl.com/yyoo8deh, 2015.

[167] S. Popov. IOTA: The Tangle, 2016.

[168] T. Prätzlich, J. Driedger, and M. Müller. Memory-restricted Multiscale Dynamic Time Warping. In IEEE Conference on Acoustics, Speech and Signal Processing (ICASSP), pages 569–573, 2016.

[169] Z. A. Qazi, J. Lee, T. Jin, G. Bellala, M. Arndt, and G. Noubir. Application-awareness in SDN. In ACM SIGCOMM conference, pages 487–488, 2013.

[170] J.-J. Quisquater and D. Samyde. ElectroMagnetic Analysis (EMA): Measures and Counter-measures for Smart Cards. In Springer Smart Card Programming and Security (E-Smart), pages 200–210. 2001.

[171] T. Rakthanmanon, B. Campana, A. Mueen, G. Batista, B. Westover, Q. Zhu, J. Zakaria, and E. Keogh. Searching and Mining Trillions of Time Series Subsequences under Dynamic Time Warping. In ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD), pages 262–270, 2012.

[172] Ransomware Attacks Are Testing Resolve of Cities Across America. [online] https://tinyurl.com/yxu9h346, 2019.

[173] J. Rauchberger, S. Schrittwieser, T. Dam, R. Luh, D. Buhov, G. Pötzelsberger, and H. Kim. The Other Side of the Coin: A Framework for Detecting and Analyzing Web-based Cryptocurrency Mining Campaigns. In International Conference on Availability, Reliability and Security (ARES), pages 1–10, 2018.

[174] J.-F. Raymond. Traffic Analysis: Protocols, Attacks, Design Issues, and Open Problems. In Springer Designing Privacy Enhancing Technologies, volume 2009, pages 10–29, 2001.

[175] F. Reid and M. Harrigan. An Analysis of Anonymity in the Bitcoin System. In IEEE conference on Privacy, Security, Risk and Trust(PASSAT), pages 1318–1326, 2011.

[176] L. Ren. Proof of Stake Velocity: Building the Social Currency of the Digital Age. [online] https://tinyurl.com/y6fwrgt8, 2014.

[177] D. Ron and A. Shamir. Quantitative Analysis of the Full Bitcoin Transaction Graph. In Springer Financial Cryptography and Data Security (FC), pages 6–24, 2013.

[178] D. Ron and A. Shamir. How Did Dread Pirate Roberts Acquire and Protect His Bitcoin Wealth? In Springer Financial Cryptography and Data Security (FC), pages 3–15, 2014.

[179] M. Rosenfeld. Analysis of Bitcoin Pooled Mining Reward Systems. arXiv preprint: 1112.4980, 2011.

[180] J. Rüth, T. Zimmermann, K. Wolsing, and O. Hohlfeld. Digging into Browser-based Crypto Mining. arXiv preprint: 1808.00811, 2018.

[181] M. Sadiku. Elements of Electromagnetics. Oxford University Press, 2014.

[182] B. Saltaformaggio, H. Choi, K. Johnson, Y. Kwon, Q. Zhang, X. Zhang, D. Xu, and J. Qian. Eavesdropping on Fine-Grained User Activities within Smartphone Apps over Encrypted Network Traffic. In USENIX Workshop on Offensive Technologies (WOOT), pages 1–10, 2016.

[183] S. Salvador and P. Chan. FastDTW: Toward Accurate Dynamic Time Warping in Linear Time and Space. In KDD Workshop on Mining Temporal and Sequential Data, pages 70–80, 2004.

[184] SecureWorks. In the Aftermath of the 'NotPetya' Attack. [online] https://tinyurl.com/y4mw9ful, 2017.

[185] J. Shawe-Taylor and N. Cristianini. Kernel Methods for Pattern Analysis. Cambridge University Press, New York, NY, USA, 2004.

[186] D. Silva and G. Batista. Speeding Up All-pairwise Dynamic Time Warping Matrix Calculation. In SIAM International Conference on Data Mining, pages 837–845, 2016.

[187] Y. Sompolinsky, Y. Lewenberg, and A. Zohar. SPECTRE: A Fast and Scalable Cryptocurrency Protocol. IACR Cryptology ePrint Archive, page 1159, 2016.

[188] C. Song, F. Lin, Z. Ba, K. Ren, C. Zhou, and W. Xu. My Smartphone Knows What You Print: Exploring Smartphone-based Side-channel Attacks Against 3D Printers. In ACM conference on Computer and Communications Security (CCS), pages 895–907, 2016.

[189] K. Soska and N. Christin. Measuring the Longitudinal Evolution of the Online Anonymous Marketplace Ecosystem. In USENIX Security, pages 33–48, 2015.

[190] M. Spagnuolo, F. Maggi, and S. Zanero. BitIodine: Extracting Intelligence from the Bitcoin Network. In Springer Financial Cryptography and Data Security (FC), pages 457–468, 2014.

[191] I. Stewart. Proof of Burn. [online] https://tinyurl.com/cjlb484, 2018.

[192] T. Stöber, M. Frank, J. Schmitt, and I. Martinovic. Who do You Sync You are? Smartphone Fingerprinting via Application Behaviour. In ACM conference on Security and Privacy in Wireless and Mobile Networks (WiSec), pages 7–12, 2013.

[193] B. Stone-Gross. The Lifecycle of Peer-to-Peer (Gameover) ZeuS. [online] https://tinyurl.com/yxevtj7y, 2012.

[194] C. A. Sunshine. Source Routing in Computer Networks. ACM SIGCOMM Computer Communication Review, 7(1):29–33, 1977.

[195] Symantec Inc. Internet Security Threat Report. [online] https://tinyurl.com/cuut5ne, 2017.

[196] Symantec Security Response Team. CryptoDefense, the CryptoLocker Imitator, Makes Over $34,000 in One Month. [online] https://tinyurl.com/q8dyqjv, 2014.

[197] Symantec Security Response Team. Petya Ransomware Outbreak: Here's What You Need to Know. [online] https://tinyurl.com/y6klw2tn, 2017.

[198] Symantec Security Response Team. What You Need to Know about the WannaCry Ransomware. [online] https://tinyurl.com/yxcrm92t, 2017.

[199] N. Szabo. Bit Gold. [online] https://tinyurl.com/yxrrapks, 2005.

[200] R. Tahir, M. Huzaifa, A. Das, M. Ahmad, C. Gunter, F. Zaffar, M. Caesar, and N. Borisov. Mining on Someone Else's Dime: Mitigating Covert Mining Operations in Clouds and Enterprises. In International Symposium on Research in Attacks, Intrusions, and Defenses (RAID), pages 287–310, 2017.

[201] V. F. Taylor, R. Spolaor, M. Conti, and I. Martinovic. Smartphone App Identification via Encrypted Network Traffic Analysis. IEEE Transactions on Information Forensics & Security, 13:63–78, 2018.

[202] Ten Arrested in Netherlands over Bitcoin Money-laundering Allegations. [online] https://tinyurl.com/hmp2ll7, 2016.

[203] J. D. Tygar and B. Yee. Cryptography: It's Not Just for Electronic Mail Anymore. Technical report, Carnegie Mellon University, 1993.

[204] U.S. District Court, Southern District of New York: Alleged Silk Road Founder Ross William Ulbricht Criminal Complaint. [online] https://tinyurl.com/y3xczd7g, 2017.

[205] V. N. Vapnik. The Nature of Statistical Learning Theory. Springer-Verlag, New York, NY, USA, 1995.

[206] P. Vasin. Blackcoin's Proof-of-Stake Protocol v2. [online] https://tinyurl.com/y2g89ec8, 2014.

[207] Q. Wang, A. Yahyavi, B. Kemme, and W. He. I Know What You Did on Your Smartphone: Inferring App Usage over Encrypted Data Traffic. In IEEE Communications and Network Security (CNS), pages 433–441, 2015.

[208] W. Wang, B. Ferrell, X. Xu, K. W. Hamlen, and S. Hao. SEISMIC: SEcure In-lined Script Monitors for Interrupting Cryptojacks. In European Symposium on Research in Computer Security (ESORICS), pages 1–20, 2018.

[209] X. Wang, S. Chai, M. Isnardi, S. Lim, and R. Karri. Hardware Performance Counter-Based Malware Identification and Detection with Adaptive Compressive Sensing. ACM Transactions on Architecture and Code Optimization (TACO), 13(1):1–23, 2016.

[210] X. Wang and R. Karri. Numchecker: Detecting Kernel Control Flow Modifying Rootkits by using Hardware Performance Counters. In Annual Design Automation Conference (DAC), pages 1–7, 2013.

[211] X. Wang, C. Konstantinou, M. Maniatakos, and R. Karri. ConFirm: Detecting Firmware Modifications in Embedded Systems using Hardware Performance Counters. In IEEE/ACM International Conference on Computer Aided Design (ICCAD), pages 544–551, 2015.

[212] X. Wang, A. Mueen, H. Ding, G. Trajcevski, P. Scheuermann, and E. Keogh. Experimental Comparison of Representation Methods and Distance Measures for Time Series Data. Springer Data Mining and Knowledge Discovery, 26(2):275–309, 2013.

[213] WannaCry Address 2. [online] https://tinyurl.com/y3u9hwuc.

[214] X. Xi, E. Keogh, C. Shelton, L. Wei, and C. A. Ratanamahatana. Fast Time Series Classification using Numerosity Reduction. In ACM International Conference on Machine Learning (ICML), pages 1033–1040, 2006.

[215] J. Yang, S. Zhang, X. Zhang, J. Liu, and G. Cheng. Analysis of Smartphone Traffic with MapReduce. In IEEE Wireless and Optical Communication Conference (WOCC), pages 394–398, 2013.

[216] L. Yuan, W. Xing, H. Chen, and B. Zang. Security Breaches as PMU Deviation: Detecting and Identifying Security Attacks using Performance Counters. In ACM SIGOPS Asia-Pacific Workshop on Systems (APSys), pages 1–6, 2011.

[217] X. Zhou, S. Demetriou, D. He, M. Naveed, X. Pan, X. Wang, C. A. Gunter, and K. Nahrstedt. Identity, Location, Disease & More: Inferring Your Secrets from Android Public Resources. In ACM Computer and Communications Security (CCS), pages 1017–1028, 2013.

Chapter A

Acronyms

BCD	Bitcoin Diamond
BCH	Bitcoin Cash
BTC	Bitcoin
BTM	Bytom
DASH	Dash
DCR	Decred
ETC	Ethereum Classic
ETH	Ethereum
LTC	Litecoin
QRK	Quark
SBTC	SuperBitcoin
SC	Siacoin
UBTC	UnitedBitcoin
XMC	Monero-Classic
XMR	Monero
XZC	Zcoin
ZEC	Zcash

Chapter B

Standard Definitions

MinMax scaler scales each feature in the range [0,1]. Specifically, given a feature x and one of its assumed value x_i the following formula is applied:

$$minmax(x_i) = \frac{x_i - min(x)}{max(x) - min(x)},$$

where $min(x)$ and $max(x)$ are the minimum and maximum value of the feature x in the dataset.

Standard scaler transforms each feature in such a way that the mean becomes zero and standard deviation becomes one. Specifically, given a feature x and one of its value x_i, the following formula is applied:

$$Z(x_i) = \frac{x_i - \mu(x)}{\sigma(x)},$$

where $\mu(x)$ and $\sigma(x)$ are the mean and standard deviation of the variable x. It is also referred to as Z-score normalization.

Standard error of a variable y is expressed as:

$$S_E(y) = \frac{\sigma(y)}{\sqrt{n}},$$

where n and $\sigma(y)$ are the number of observations and standard deviation of the variable y.

Margin of error is the range of values above and below the sample mean for a given confidence interval. It is calculated as:

$$z * S_E(y),$$

where z is the coefficient for the selected confidence level. E.g., z is 1.96 for 95% confidence interval.

Accuracy measures how often the classifier makes the right prediction defined as the ratio between the number of hit and the number of predictions.

Precision quantifies the ability of a classifier to not label a negative example as positive. It is computed as the ratio of the number of true positives and the total number of instances labeled as positives.

Recall defines the probability that a positive prediction made by the classifier is actually positive. It is computed as the ratio of the number of true positives and the total number of positives in the set.

F_1 score is a single metric that combines both precision and recall via their harmonic mean:

$$F_1\ score = 2 \times \frac{precision \times recall}{precision + recall}$$

Chapter C

Seed Bitcoin Addresses for Ransomware

#	Address	Source(s)
1	135N2nfAkextd6E25quXpM98qLSi2BccCb	[67]
2	1AEoiHY23fbBn8QiJ5y6oAjrhRY1Fb85uc	
3	18iEz617DoDp8CNQUyyrjCcC7XCGDf5SVb	[67, 68]
4	1KP72fBmh3XBRfuJDMn53APaqM6iMRspCh	

Table C.1: CryptoLocker

#	Address	Source(s)
1	19DyWHtgLgDKgEeoKjfpCJJ9WU8SQ3gr27	[196]
2	1EmLLj8peW292zR2VvumYPPa9wLcK4CPK1	

Table C.2: CryptoDefense

#	Address	Source(s)
1	1PoebUjR5pdH88tc9ECQ1PCLaCrtPnG9fm	[69]
2	128pJdREzcR6xorYPQAPzGf8RwMQjRBzDt	[73]
3	15WUYqKerTtxi4rUEmnakw5gRMkr3nZCQd	[74]
4	1L66AcnbuZkYjs8eE6uVbTUxmorHYGKxFJ	[75]
5	16REtGSobiQZoprFnXZBR2mSWvRyUSJ3ag	[76]
6	16Z6sidfLrfNoxJNu4qM5zhRttJEUD3XoB	[77]
7	12LE1yNak3ZuNTLa95KYR2CQSKb6rZnELb	[78]
8	1JYYzNHDaGC7noiE4eKatuYA4AThqVocDd	[79]
9	1BhLzCZGY6dwQYgX4B6NR5sjDebBPNapvv	[80, 84]
10	16yd1Wj2NZa2uLZ6W4UDCDJ2Ttw92uFaT7	[70, 84]
11	1LGnuv6KX9SXB8eM72dnBAcECeaC8Z2zje	[71, 84]
12	1L7SLmazbbcy614zsDSLwz4bxz1nnJvDeV	[72, 84]
13	19yqWit95eFGmUTYDLr3memcDoJiYgUppc	
14	16N3jvnF7UhRh74TMmtwxpLX6zPQKPbEbh	
15	1ApF4XayPo7Mtpe326o3xMnSgrkZo7TCWD	
And, 27 other distinct addresses that are listed on [84]		

Table C.3: CryptoWall

#	Address	Source(s)
1	1MrKJhiECV3RufrY1dSybSXRCwSw11Co6i	[90]
2	1C8yA7wJuKD4D2giTEpUNcdd7UNExEJ45r	[91]
3	166vHLnGB1pCQGxdBkRiMkHW5WGQDbsw6s	[92]
4	1BA48s9Eeh77vwWiEgh5Vt29G3YJN1PRoR	[93]
5	18mfoGHSfe9h145e8djHK5rChDTnGfPDU9	[94]
6	16hHkyuzCDRFzoejVuqajqrnbmKHSmEfQM	[95]
7	1382JAg5xbQv7QNwq1svDeyw6ELtNCmujG	[89]
8	1KXw7aJR4THWAxtnxZYzmysdLXVhLfa97n	

Table C.4: DMA Locker

#	Address	Source(s)
1	13dN96pRTQDhpWRqKyLTbgRxeTN52p2CqY	[151]

Table C.5: Mischa

#	Address	Source(s)
1	1BAdEKq6zE1JDL8g2pA1MDRHbW1wvYCWhT	[107]
2	1MGnopAa6MAGjUpCEmRiSAcVKZNB6n8gnR	[108]
3	17xV74Hp2zNR74yG3AJvPpNMchPJHm2iUo	[109]

Table C.6: GoldenEye

#	Address	Source(s)
1	1Mz7153HMuxXTuR2R1t78mGSdzaAtNbBWX	[159, 197]

Table C.7: NotPetya

#	**Address**	**Source(s)**
1	1PGAUBqHNcwSHYKnpHgzCrPkyxNxvsmEof	[124]
2	1Lhgda4K77rFMTkgBKqmsdinDNYYVbLDJN	[125]
3	1KGusS7xB9hnqZQdCZ1G8Tno16RfTS95ey	[126]
4	1KPPqHpd8Z9S6pQH1qVovzyejyfDMghp4u	[127]
5	1J9PMCpbrnicZoBUdyuNBwi4QvXwq6Korq	[128]
6	16hhyeg7WMh4Go7JqNKRwmD95bRd4aenwz	[129]

Table C.8: KeRanger

#	**Address**	**Source(s)**
1	13AM4VW2dhxYgXeQepoHkHSQuy6NgaEb94	[65, 213]
2	12t9YDPgwueZ9NyMgw519p7AA8isjr6SMw	
3	115p7UMMngoj1pMvkpHijcRdfJNXj6LrLn	

Table C.9: WannaCry