5-2024

Monero: Powering Anonymous Digital Currency Transactions

Jake Braddy

University of Nebraska Omaha

Monero: Powering Anonymous Digital Currency Transactions

Jake Braddy

Professor Grispos

May 17th, 2024

Table Of Contents

Abstract

Cryptocurrencies rely on a distributed public ledger (record of transactions) in order to perform their intended functions. However, the public's ability to audit the network is both its greatest strength and greatest weakness: Anyone can see what address sent currency, and to whom the currency was sent. If cryptocurrency is ever going to take some of the responsibility of fiat currency, then there needs to be a certain level of confidentiality. Thus far, Monero has come out on top as the preferred currency for embodying the ideas of privacy and confidentiality. Through numerous cryptographic procedures, Monero is able to obfuscate or obfuscate the following: the address of the sender and recipient, the amount that was sent, and the origin of the sender. Monero is sent by taking the private ownership keys for a grouping of Monero and issuing a command to the network to reassign them to a new public ownership key. Thanks to the advent of asymmetric cryptography, there is no current computationally feasible way to link a public key to a private key and unique sub-keys can be generated at any time to allow for further obfuscation. Additionally, Monero uses a novel implementation of ring signatures which allows you to hide the proof of ownership among a group of users. Currently, transactions are signed with 11 potential true sender's signatures and the only information an outsider can reveal is a guarantee that one and only one of those signatures did own the private key for the currency being sent. Additionally, Monero uses advanced mathematics for its routing protocol to ensure it is nearly impossible for an outsider to identify the original source IP that initiated a transaction merely by participating in the network. Finally, there is amount confidentiality, which is possible thanks to a 2018 research project from Stanford called Bulletproofs. Bulletproofs enable an efficient method for conducting a zero-knowledge proof that allows the amount sent to never be revealed and yet outsiders are still able to verify that the funds are legitimate. Even though Monero is open source, hence there is a wide array of contributors and literature, there are still some questions left unanswered after investigating all of the popular sources. First, are there any traces left on a PC that would de-obfuscate a transaction? Second, would an internet service provider (or other party with widespread access to internet infrastructure) be able to identify the source of a transaction? Finally, it is widely known that Monero uses algorithms that are not quantum resistant, so how will quantum computers affect the privacy measures currently in place?

Introduction

In a traditional financial system, there is a reliance on trusted third parties to conduct a transaction whether this is a government assigning value to a currency, a bank holding the funds, or a payment processor moving funds. Cryptocurrencies take these norms and instead move the trust to the math and codes which allow the secure movement of an encrypted value. Rather than rely on an institution with unknown internal practices we can instead entirely rely on publicly available and tested mathematical functions and cryptographic structures to conduct the same transactions. Cryptocurrencies work by continuously signing and overwriting digital signatures on an encrypted asset. These signings are carefully checked and balanced by vast networks of crowdsourced computers that dedicate their resources in return for a chance at earning commissions. However, the exact implementation of the signatures, checks, and interconnections can vary widely per coin from forcing users to use a trusted third party to all manner of algorithms for routing, signatures, and verification. In fact, there can even be multiple, separately named coins all operating with the same underlying code but operating on a separate network. Therefore, to conduct a proper security analysis and review, this paper will focus primarily on Monero with shorter examples of Bitcoin, and the underlying Cryptonote protocol, for simplifications and comparisons.

Cryptocurrencies have maintained relevance even after their period as a fad technology and are actively used on a daily basis. Although the exact value does not matter, Bitcoin currently sits at a 1.25 trillion-dollar market cap and Monero has a 2.5 billion-dollar market cap (coinmarketcap, 2024) (Market cap is not representative of true value it just takes the current price of one coin multiplied by the number of coins in circulation). Bitcoin works as a great case example for explaining the foundational concepts that allow Monero to function, but our core goal will be to develop a complete understanding of the privacy mechanisms used by Monero. Monero was developed to create a true digital equivalent to cash where two parties can conduct a transaction without a need for a third party. Initially, this was an expectation of Bitcoin as well, but, due to numerous privacy/security failings, the coin quickly fell short of these expectations. The problem lies in the necessity for individuals to verify that a transaction is legitimate in terms of origin, recipient, and amount without revealing this information to perceptive onlookers. Consider this, would you want your bank account, credit card history, or any other financial information to be public knowledge? With coins, like Bitcoin, the sender, recipient, and amount of all transactions is completely public knowledge. The creators and maintainers of Monero examined these problems and created a solution that utilizes novel security and cryptographic concepts and techniques to encrypt the bulk of information.

This document will serve as a guiding point for developing further research on Monero and will be structured as follows. First, related work will be introduced, this section includes the majority of the popular/notable papers that have been used to train the Monero community at large. Additionally, this section will contain the whitepapers and original publications for any underlying protocols when possible. This section will then conclude with a few grievances with the popular literature that will be explored in the remaining sections. Following the related works will be a summary of the research

method used to produce any results, which includes, but is not limited to, the software and hardware utilized in my testing. The next section, ‘Findings’, will take the results of the research method and attempt to extrapolate any new information as it pertains to the general grievances assessed in the related work. To round this out, the topic of quantum cryptography will be introduced as it is paramount that this technology is discussed and adequetaly addressed while there is still time. To wrap this all up, the conclusion will contain some ideas for future work that are likely to benefit the Monero project as a whole, along with a discussion and summarization of all the work presented.

Related Work

Before we can dive into Monero's advanced cryptosystems, we first need to understand the blocks on which it was built. These items are the generalized concepts necessary for a cryptocurrency and may contain inherent weaknesses upon which Monero iterates and adapts. This section will pull the bulk of its information from the Mastering Monero book written by SerHack along with the Monero community at large (Serhack, 2019). In short, nodes are connected via a peer-to-peer network where they share and modify a file that acts as the source of truth; in the case of cryptocurrencies, this is a list of transactions known as a distributed public ledger. Each node can append to or read from the file, but each change must be mathematically verified before all nodes will accept and record the modification. Nodes can compare their ledgers to one another to correct discrepancies using majority rule. However, new transactions must undergo rigorous verification by miners. Specifically, miners are users who dedicate their computing power in return for the chance to earn currency. The rewards are attributed based on a Proof of Work (PoW) algorithm where the fastest/luckiest solver of a random number guessing competition is allowed to submit their solution to the verification problem. This participant is chosen semi-randomly to dissuade participants from submitting wrong/malicious answers that would later be unverified in the next round of mining. Essentially, to perform an attack where transactions are maliciously manipulated an attacker would need to be the fastest solver of a random number guessing competition many times in a row to pass every round of mathematical variation (Monero requires transaction blocks to be verified ten times before they are considered official). Rewards are generated by a combination of transactions fees and a potential amount of reserve/new currency (Monero will always payout reserve/new coins, but other coins, like Bitcoin, will not). Miners can work together as a group, known as pools, to increase the likelihood of payouts, but, in return, they split the rewards.

There are concerns of attacks known as the 51% attack where a single participant, or group of bad actors, possess a majority of the network's computing power. This party would then be able to alter the blockchain and pass it off the correct answer with some degree of certainty. Monero does in fact suffer from this same vulnerability, but makes large pools much less discouraged, and offers P2P alternatives. Additionally, Monero uses an algorithm that favors CPU heavy systems rather than the GPU or ASIC (application specific integrated circuit) heavy networks of Bitcoin, and other top currencies. RandomX favors certain CPU focused operations that are still relatively efficient even on older CPUs, and thus allow a wide array of miners to be competitive in Monero. Specifically, "RandomX uses random code execution (hence the name) together with several memory-hard techniques to minimize the efficiency advantage of specialized hardware" (Tevador, et al., n.d.).

The primary issues with other cryptocurrencies, like Bitcoin and Ethereum, lies in the fact that their verification procedures require a significant amount of transaction information to be public knowledge (sender, recipient, and amount transacted). This means that any individual with an internet connection (free blockchain explorer tools are widely available) can view the full transaction history for any user if their wallet address is ever tied to their identity. Monero iterates on these challenges by introducing a variety

of cryptographic security measures such as ring signatures, zero-knowledge proofs, key derivation functions for unique wallets, and a clever implementation of one-time outputs to encrypt most of the transaction data.

One of the original teachings for Monero contributors is Kurt Alonso's master's thesis, 'Monero - Privacy in the Blockchain. In this work, they relay that Monero practices separation of keys to increase security "Unlike Bitcoin, Monero users have two sets of private/public keys" ... "key k1 will be called view key whereas k2 will be the spend key" (Alonso, 2018). Even after this separation, the keys are not directly utilized for a transaction, rather, they are combined with a random number chosen for that transaction to form a one-time key. The random number is chosen only by the participants in the transaction and is similar to a Diffie-Helman exchange. This unique value is then used twice, first to sign the key image (the unique value representing all coins being spent) and it is then included in the ring signature. A ring signature is a method for combining many keys together such that it is not possible to discern which one is the true key. There are many types of Ring signatures and Monero's implementation combines randomly chosen potential keys from the chain itself and the true one-time key mentioned earlier. From there, an outside participant is only able to determine that there is one, and only one key, true key in the mix.

Kurt Alonso, Sarang Noether, and Koe later released an updated and more refined version of Alonso's original thesis under the title "Zero to Monero: Second Edition" released in 2020. This paper now serves as the de facto literature for preparing a community member to understand the mathematics and implementation of the Monero network. For those with backgrounds in mathematics, it will be the most in-depth literature available for mastering the fundamental cryptography. However, the mathematics itself largely extends beyond my capabilities, so I had to rely on intuition and their word-based summarizations and further research would be required if I were to discuss the exact implementation of such mathematical protocols.

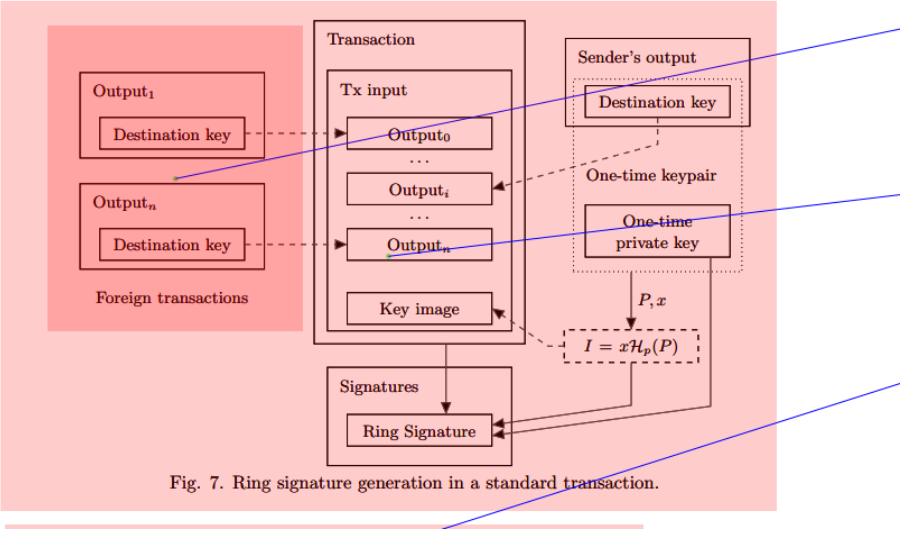

Fig. 7. Ring signature generation in a standard transaction.

Fig 0. Pictures from the underlying CryptoNote code that gave rise to Monero. This is the annotated version with comments from a Monero developer. (Van Saberhagen, et al., 2013)

Monero relies on the contributions of many works together to create a layered privacy landscape of mitigations and avoidance. As previously mentioned, Monero functions on a peer-to-peer network where users directly contact each other. Users/Nodes complete this task by each keeping their own records of the ~100 most recent senders of any messages they receive and maintaining this list allows them to select random recipients when the time comes. Messages are not simply sent to a single user on this list, rather, they are routed using a protocol referred to as Dandelion++. This algorithm received its name due to the propagation method of the underlying messages: A phase of single transfers (like a stem) until a certain number of transfers then a broadcast phase (like the seeds). However, this is the simplified explanation for the prior Dandelion (no ++) algorithm that was discovered to have statistically significant vulnerabilities in propagation which was then replaced by the more random Dandelion++ variant. A brief summary of the underlying dandelion protocol from the Dandelion++ paper is as follows (Fanti, et. Al., 2018). Rather than the stem phase following a simple line graph, it instead uses a random 4-regular graph by also selecting two random, outer-edge neighbors to potentially forward the message. The protocol will then forward all transactions across a random path of inner edges to the selected outer edge. Dandelion++ also changes this random graph every 10 minutes, to prevent excessive information leakage. In short, this protocol allows us a reasonable degree of certainty that an outside user could not backtrack the message to determine its original point of origin.

Finally, the remaining literature of note is the Bulletproof paper from Stanford (Bunz, Benedikt, et al., 2018). Bulletproof is a logarithmically scaling (very efficient) method of creating zero-knowledge proofs for confidentiality. This relies on the concept of a Pederson commit, and, in very simple terms, is a way of proving that the sum of the inputs equals the sum of the outputs without revealing any information about the inputs

and outputs. The integrity of mathematics is outside of the scope of this paper, and they even take quantum into consideration.

After reading through the most popular Monero resources, Zero to Monero 2nd edition, Mastering Monero, and the annotated cryptonote whitepaper (along with the numerous corollary papers for the underlying protocols), I still had some remaining concerns regarding privacy that I wanted to personally verify. These concerns fall into two categories: what information could be recovered with local access to a machine that previously ran a wallet but entered a powered-off state, and what information can be assessed purely from network traffic when it leaves the internal network? Additionally, Mastering Monero, which is the most comprehensive resource, did not delve into the topic of network security due to the author's belief a new update would resolve all issues. However, this update was later scrapped, and network information is not masked by default. There were prospects to connect Monero into TOR or a similar relay system, but these are still not implemented, and their progress is unclear. In a similar light, Kurt's original master's thesis, "Zero to Monero: Second Edition", and Mastering Monero all fail to include the word quantum even once in their notable papers.

Research Method

Local Access

To begin with, the Monero community highly advises utilizing a known good wallet with a default configuration, so all experiments will be done on the most up-to-date version of the GUI wallet (Current Version: 0.18.3.2 - Fluorine Fermi) at the time of writing. From here our first goal is to test what traces of transactions can be recovered from simply viewing the contents of the hard drive. To accomplish this documentation, I created a fresh Windows 11 Pro VM and let it fully update (As of testing this was 23H2 OS build 22631.3296) . From there created a baseline Wireshark Control.PCAP file was generated (30 minutes of activity) to allow for a control group for later experiments, more info on this in the next section. From there, the Monero GUI wallet was installed while ProcMon (a Windows utility that allows us to monitor all items accessed by a program) was running. From there, a new wallet was created and the ProcMon log was ended once syncing started. Although the underlying hardware should be irrelevant to our test results, the VM was created on VMWare workstation player 17.5.1 build-23298084, and it is running on AMD Ryzen 7800x3d with 16GB of virtual memory allocated.

Network Eavesdropping

The next step is to analyze the network communications done by the wallet using the Wireshark program, specifically, version 4.2.4 (v4.2.4-0-g1fe5bce8d665) . Wireshark is a tool that allows us to capture the network packets containing all information that enters or leaves a device from its network interface card. For our purposes, many captures were generated as even a fresh install is incredibly noisy as it attempts to enumerate local devices and communicate with Microsoft. As mentioned previously, we already generated a control group (Control.PCAP) before ever installing the GUI wallet. Then, a new Wireshark BaseSync.PCAP file was generated (30 minutes of activity) to identify normal syncing traffic shortly after the wallet install. Then, to save time and data, most of the blockchain was manually imported and, to counteract this measure, another file was generated BaseSync2.PCAP (30 minutes of activity) to capture the final sync. To continue this mapping, another Wireshark DATA.PCAP file was generated (30 minutes of activity) to capture normal traffic when the chain is fully synced and up to date. At this point, we have the following files:

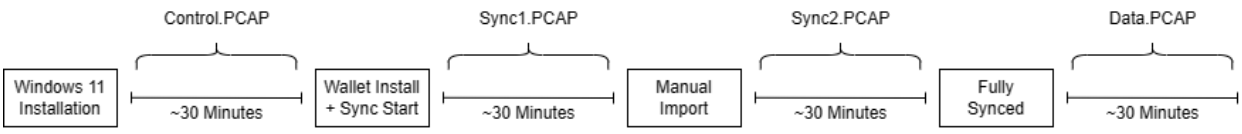

Figure 1. Created by author and Logan Mears

Theoretically, the sync traffic should be nearly identical to the baseline syncing because it's just syncing the new blocks every two minutes. Receiving traffic falls into this same boat as well where, to all other participants, it is no different than any other sync. However, sending is the portion that we are truly concerned with as it requires a modification done by the end user to be uploaded. Does this then mean that we can see any noticeable difference between the download/sync operation and the upload/send operation? For this, I sent a transaction from the test wallet while monitoring traffic and captured from before clicking send through when it appears as an unconfirmed transaction on the recipient wallet (took about 45 seconds). This transaction and its corresponding capture was titled Send1.PCAP. This was then repeated, but with a shorter window (from just the confirm screen until unconfirmed account balance) which resulted in a packet capture of just under 30 seconds. Note that this means transactions become viewable by the receiving wallet even though it is either unverified or, at most, single-verified (new blocks are approved and added roughly every two minutes and 10 blocks must be added before a transaction is completely verified). The other transaction and corresponding capture were titled Send2.PCAP.

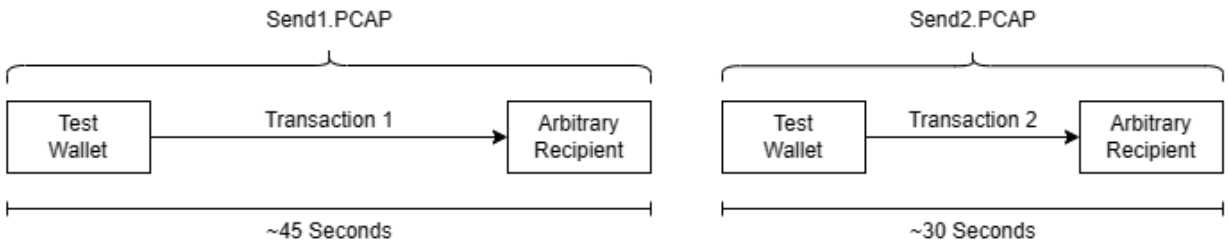

Figure 2. Created by author and Logan Mears

Findings

Local Access

After creating all necessary logs, I read through the entire ProcMon log, from wallet creation and wallet installation, for any remnants that involved the creation or modification of values. Potential items of interest include sources of RNG, temp files, registry-based storage, and any items that will persist, except the wallet file itself and the necessary program files for running the GUI. However, as largely expected, there were no clear remnants that would expose information linking a user to their unique wallet address/keys. However, I will note that by using the GUI wallet the user is trusting that the code utilized does not decrease security. As demonstrated during a Monero conference in 2019, it is possible to visualize the Monero blockchain (Krawiec-Thayer, 2019). Using these techniques, any changes that make transactions look different (such as using specific, non-default fee values) makes it possible to reduce the anonymity of a transaction. Beyond that, further analysis of the public repository is recommended to ensure no bloat nor bug would decrease security. However, the wide range of languages and contributors made this an unfeasible task for this paper's purposes.

Network Eavesdropping

Using Wireshark, we can see the communication logs necessary for Monero-oriented communications. By comparing the send logs to the control logs, we can then remove any noise (information not related to our Monero transfer). In these investigations, it is clear that Monero packets follow a specific structure with a well-defined header. Further examination reveals that there appears to be two header types, a long message originating from our host beginning with 005056e79c46000c and short responses from the recipients with a header beginning with 000c2968ad0b0050. Within these blocks, we can identify other form fields that seem to stay consistent within a particular send but vary between send1 and send2. One such item is the tx field, which seemed promising at first, but, on further inspection, did not correlate with the full transaction information available to me as the sender. Likely, there are even more layers of mathematical functions being applied which are largely absent from the mainstream documentation. Additionally, it is important to note that reading information directly from packets is ill-advised without knowing the specific structure of the item being analyzed. In this light, it would be beneficial if tools were available that defined the structure of the local block and the typical packet to make analysis and comparisons easier.

In the absence of these tools, here is what we know; every transaction must produce a unique key image that denotes the combination/proportion of the individual coins being transferred. Additionally, the recipient must be denoted, but there is a Diffie-Helman-esque modification made to the recipient id to combine it with a random value making recovery nearly impossible to identify or recover. Purely from log analysis, there are still a few more interesting items of note:

1) Our node is reaching is sending out an unsolicited large packet with a special header.

2) Nodes may then reply, and their messages utilize a different structure.
    a. Typically, nodes will reply many times with slightly different data, but the same header.
3) During this time, it is possible to receive unsolicited large header packets from other nodes.
4) Going back to our sync files, we can see that it appears like the small packets are being used as requests for and confirmations of a block. Whereas the large packets are the contents of a block – transaction(s).

From this, the only clear identifier of a transaction is that the block is sent unsolicited to other nodes. However, due to the dandelion++ protocol, it is assumed to be impossible to track/estimate the route a transaction will take as it propagates across the Monero network.

The unsolicited nature of the transaction does appear to provide some uniqueness/identifiability for tracking Monero, but this would likely be insufficient by itself to form any true analysis. However, that's not to say this information is useless; assuming the transaction is not modified per-hop, it is still a unique value that follows a specific behavior. If a party were able to capture all, or a majority, of the Monero traffic, it could be possible to follow the timestamps to identify the original unsolicited sender of the structured, but unique packet. If multiple transactions are performed and a specified host (e.g. IP address) continuously ranks as an early/initial sender of the message, then the odds that the host is the true sender skyrockets. However, this style of attack would be defeated by a simple VPN, TOR, or any form of peer-to-peer encryption. Note: it is possible that there is peer-to-peer encryption or per-hop modifications in place, but it is nearly impossible to tell without the aforementioned analysis tools and a larger scope of experimentation. Therefore, these results are inconclusive at best but serve as a call to arms for proper experimentation to be performed on whether it is acceptable to continue leaving the network layer potentially exposed without the advent of a security layer (like Mastering Monero's assumed implementation of 'Kovri' or the WIP TOR adoption).

Discussion of Quantum Impact

Many warn that quantum computing will break typical cryptographic algorithms within the next 15 years or so, like the 256-bit elliptical curve cryptography (ECC) that serves to link Monero's public/private spend and view key. Currently, IBM, one of the leaders in quantum computing, has their top quantum chip sitting at 133 qubits of performance (IBM, 2024). To quantify the time and qubit requirements for breaking ECC, Daniel Litinski from Palo Alto calculated the time necessary for algorithms functioning with 1152 qubits (Litinski, 2023). From this, we can extrapolate that the first hurdle for quantum is an 8.66165414x increase in qubits, or, from the perspective of silicon-based innovations, just over 3 doublings. Of course, there are other considerations like the clock speed, amount of noise, the efficiency of the implemented algorithm, and other physical constraints (like the number of Toffoli necessary), but this at least provides a frame of reference for the quantum estimates. Additionally, multiple implementations are suggested that tie in some of these considerations to create time-to-crack estimates ranging from 160 days per key on the low end to just 10 minutes with more performant hardware speculations. However, the real question is how this will affect the longevity of data privacy for blockchains currently sitting on hard drives. What are the effects when this longevity is broken by the advent of powerful quantum computers?

Our first concern with quantum is that Monero relies on ECC for key generation/linkage specifically, "Monero uses a particular Twisted Edwards elliptic curve for cryptographic operations, Ed25519, birational equivalent3 of the Montogomery curve Curve25519… Elements…are 256-bit integers" (Alonso, 2018). This algorithm is used to tie the public/private view key and public/private spend key and is not quantum resistant. This means that a quantum user would be able to generate the corresponding private key if either public key is actually used publicly. However, Monero does have some protection against these measures in the form of one-time outputs and ring signatures as described in the Related Works section. Additionally, it is unclear how this will affect the ability to tie sub-addresses to addresses.

> "While Alice can calculate the public key for the address, she can not compute the corresponding private key, since it would require either knowing Bob's second private key, or solving the discrete logarithm problem for KB2 = kB2G, which we assume to be hard" (Alonso, 2018).

Additionally, Monero relies on the Bulletproof procedure for protecting amount confidentiality which, according to their own whitepaper; "the commitment is now only computationally hiding, but that switching to quantum secure range proofs is possible." (Bunz, et al., 2018). Currently, no literature clearly describes which version is currently implemented in Monero. However, judging by the phrasing in the original Bulletproof work, it seems like the quantum-resistant variant is still a work in progress and not yet used.

No available literature on how quantum affects the Dandelion++ routing algorithm for the P2P network. Quantum is already very good at solving mapping-related problems thought to be impossible for traditional computers, such as the traveling salesman

problem, so what does this mean for Monero? If an outside user can identify the IP address that first broadcasted a transaction, then it does not matter whether their wallets are secure, the anonymity is largely removed.

There are entire papers dedicated to the quantum prospects, but there are inconsistencies in these findings. For instance, as already mentioned, Bulletproofs creators considered the possibility of a quantum attack, yet no mention of this is included in popular works instead simply labeling it as vulnerable (Kearney, et al., 2021). However, they do rightly declare that, while many portions of Monero are each separately vulnerable, the combination of protocols makes it significantly more difficult for attackers as it would require a quantum solution to all transactions included in the ring. Additionally, there appears to be some disagreement as a user who concatenated their own research to create suggestions for the Monero source code (Mitchellpkt, 2020) rates the vulnerabilities quite differently than Kearney, et al. They disagree on whether Monero is truly vulnerable to public/private key pairings thanks to the Diffie-Helman like exchange used for every transaction. Overall, I believe there to be a more significant amount of community research before we know with significant certainty whether Monero can be trusted in a quantum future.

Conclusions and Future Work

Monero offers many opportunities; as a financial instrument, as a means to transfer funds anonymously, as a pillar of the open-source community, as a cryptographic pioneer, as a blockchain revolutionizer, and many more. All sorts of people can interact with Monero and walk away with value, even though Monero requires none in return. As a disciple of cyber security, Monero has taken my cryptographic skills to the next level by providing entirely open examples of how to protect financial information. Yet, Monero provides value to all, regardless of intent, by allowing a way for those bad intentions to perform illicit transactions. This makes Monero even more important, as finding a critical vulnerability could lead to transactions becoming plaintext. While, presently, there do not seem to be any major holes in Monero's security, the scattered documentation and stacked, complex protocols make visualization and complete knowledge inviable for most. A range of open-source visualizations that would better allow the participants to understand the data in rest and in transit as it lives on their machine would be beneficial for the community. Thisd could be accomplished in a few ways depending on the needs of the user and developer. Firstly, there is a possibility of using the mdb_stat command within LMDB utils to iterate through the binary data on the LMDB database. Another possibility is querying the Monero Daemon itself on any locally installed wallet using remote process calls in the JSON format. However, the Daemon is not intended to be used for this purpose, so it is unclear how well it would work. The final possibility I identified is querying the API of a Monero blockchain explorer website (e.g. https://localmonero.co/blocks/api or https://xmrchain.net/). These tools, their alternatives, and many other resources can be found at: https://www.getmonero.org/resources/tools

Even though Monero's current state seems exceptional, the possibility of a time-bomb zero-day that reveals a large amount of transaction data prevents me from fully endorsing Monero. It is important that people can conduct their business (financial and otherwise) privately, without being tracked, and Monero provides an essential service in that respect. Even though Monero's security is not perfect, among security researchers, it is often said there is no such thing as a secure system unless it's off, buried, destroyed, or many other proverbial analogies. From all steps of the transaction being encrypted separately to ensuring there will always be a small amount of new currency minted to promote miner independence, the Monero developers have truly considered the possibilities. The Monero community successfully identified the gaps in popular cryptocurrencies and leveraged this to create a truly secure (for now) product. Proper security is established through many layers of checks and encryption to ensure that each step only has as much information as is necessary to perform the necessary tasks; to verify that funds exist, the miner doesn't need to know the origin, just that the hashes are correct and that the inputs minus the outputs equals zero. Additionally, although it was only mentioned in passing, the greatest strength of Monero is that all transactions should look identical from an outside user's perspective. It is not that each transaction is completely unrecognizable, rather, the focus is to make all transactions indistinguishable from any other transaction. For any person who seeks to explore the world of Monero, remember to use a trusted, default configuration and avoid features that would make your transaction or instance stand out. For instance, there were no logs left on the test PC,

however, that is because the default log level is 0 on a 0-4 scale. If a user accidentally or maliciously enabled non-default logging, then information would have been left in the GUI wallet's log file (information like recent balances and snippets of transaction history).

In summary, Monero appears largely secure and addresses many of the main security concerns with viewing cryptocurrencies as a viable alternative to cash and credit. Monero achieves its security status by applying individual layers to protect each vulnerability and does so while retaining its status as entirely open source. However, this does not mean there is no room for vulnerabilities currently undiscovered. By the very nature of blockchain technology, one can always assume that the transaction data that currently exists will remain in its current security state on a storage medium somewhere. This means that any zero-day (unpatched and previously unknown) vulnerabilities could spell the end of security for existing transactions. Developers must begin implementing quantum reduction/prevention measures today to ensure that confidentiality and integrity remain in their products long into the future. Additionally, Monero, while open source, still has many areas that have not yet been entirely opened to the public in terms of availability of literature and tools. Even though Monero has room to grow, it still serves as a proper benchmark for applied cryptography, blockchains, and digital currencies.

Works Cited

Alonso, Kurt M., and Jordi Herrera Joancomart´ı . Monero Privacy in the Blockchain v1.0. 2018. Universitat Aut`onoma de Barcelona, https://eprint.iacr.org/2018/535.pdf.

Alonso, Kurt M., et al. Zero to Monero: Second Edition. Second Edition, 2020, https://www.getmonero.org/library/Zero-to-Monero-2-0-0.pdf.

Bunz, Benedikt, et al. Bulletproofs: Short Proofs for Confidential Transactions and More. 2018. Stanford University and University College London, https://web.stanford.edu/~buenz/pubs/bulletproofs.pdf.

CoinMarketCap “Cryptocurrency Prices, Charts And Market Capitalizations.” CoinMarketCap, https://coinmarketcap.com/. Accessed 16 May 2024.

Fanti, Giulia, et al. Dandelion++: Lightweight Cryptocurrency Networking with Formal Anonymity Guarantees. SIGMETRICS’18 Abstracts, Irvine, CA, USA , 18 June 2018, https://dl.acm.org/doi/pdf/10.1145/3292040.3219620.

IBM. Technology | IBM Quantum Computing. May 2024, https://ibm.com/quantum/technology.

Litinski , Daniel. How to Compute a 256-Bit Elliptic Curve Private Key with Only 50 Million Toffoli Gates. Palo Alto, June 2023, https://arxiv.org/pdf/2306.08585.

Mitchell Krawiec-Thayer. Visualizing Monero: A Figure Is Worth a Thousand Logs. Directed by Mitchell Krawiec-Thayer, 2019, https://www.youtube.com/watch?v=XIrqyxU3k5Q

Mitchellpkt. Identifying Practical Post-Quantum Strategies for Monero. Oct. 2020, https://github.com/insight-decentralized-consensus-lab/post-quantum-monero/tree/master.

SerHack, and Monero Community. Mastering Monero: The Future of Private Transactions . First Edition, 2019, https://masteringmonero.com/book/Mastering%20Monero%20First%20Edition%20by%20SerHack%20and%20Monero%20Community.pdf.

Tevador, et al. RandomX. Github, https://github.com/tevador/RandomX.

Van Saberhagen, Nicolas, and Monero Research Lab. “Crypnotenote Annotated Whitepaper.” Getmonero.Org, Monero Research Lab, 17 Oct. 2013, https://www.getmonero.org/resources/research-lab/pubs/whitepaper_annotated.pdf.