Responsible Vulnerability Disclosure in Cryptocurrencies

RAINER BÖHME*, Universität Innsbruck
LISA ECKEY, TU Darmstadt
TYLER MOORE†, The University of Tulsa
NEHA NARULA, MIT Digital Currency Initative
TIM RUFFING, Blockstream
AVIV ZOHAR, Hebrew University Jerusalem

Interest in cryptocurrencies has surged in recent years. Today thousands of currencies are in circulation, collectively worth hundreds of billions of dollars. Software vulnerabilities have also proliferated, which poses new and unique challenges to the ecosystem as it has developed. This review article explains what is different about vulnerabilities and responsible disclosure in cryptocurrencies, identifying key problems and opportunities for research and development. Selected case studies of vulnerability disclosures are presented. We draw lessons and pose open questions that can inform the responsible disclosure debate in cryptocurrencies and beyond.

ACM Reference Format:
Rainer Böhme, Lisa Eckey, Tyler Moore, Neha Narula, Tim Ruffing, and Aviv Zohar. 2018. Responsible Vulnerability Disclosure in Cryptocurrencies. 1, 1 (July 2018), 14 pages. https://doi.org/10.1145/1122445.1122456

1 INTRODUCTION

Despite the focus on operating in adversarial environments, cryptocurrencies have suffered a litany of security and privacy problems. Sometimes, these issues are resolved without much fanfare following a disclosure by the individual who found the hole. In other cases, they result in costly losses due to theft, exploits, unauthorized coin creation and destruction. These experiences provide regular fodder for outrageous news headlines. In this paper we focus on the disclosure process itself, which presents unique challenges compared to other software projects [15]. To illustrate, we examine some recent disclosures and discuss difficulties that have arisen.

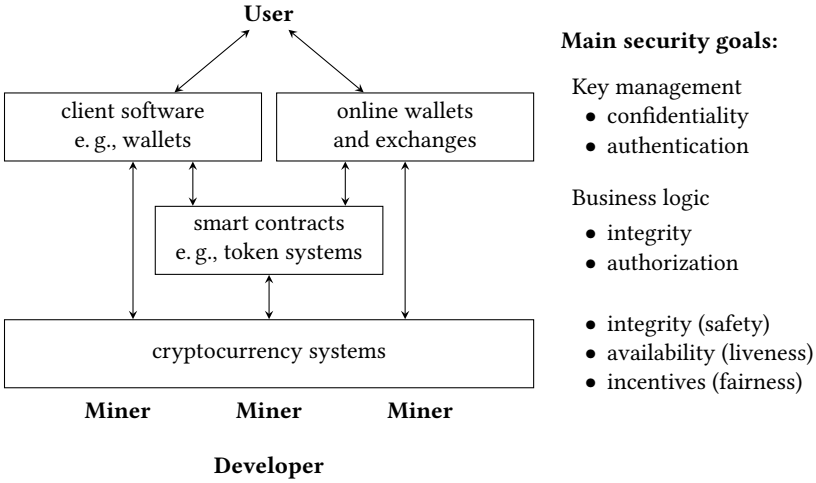
The cryptocurrency ecosystem. While Bitcoin is the best known, more than 2,000 cryptocurrencies are in circulation, collectively valued at nearly $225 billion as of October 2019 [6]. Fig. 1 conceptualizes the landscape as a stack. While the details differ, at the lowest level, each cryptocurrency system is designed to achieve common security goals: transaction integrity and availability in a highly distributed system whose participants are incentivized to cooperate [38]. Users interact with the cryptocurrency system via software "wallets" that manage the cryptographic keys associated with the

*Authors listed alphabetically.
†Corresponding author.

Authors' addresses: Rainer Böhme, rainer.boehme@uibk.ac.at, Universität Innsbruck; Lisa Eckey, lisa.eckey@crisp-da.de, TU Darmstadt; Tyler Moore, tyler-moore@utulsa.edu, The University of Tulsa, 800 S. Tucker Dr. Tulsa, Oklahoma, 74104; Neha Narula, narula@mit.edu, MIT Digital Currency Initative; Tim Ruffing, crypto@timruffing.de, Blockstream; Aviv Zohar, avivz@cs.huji.ac.il, Hebrew University Jerusalem.

Manuscript submitted to ACM

Fig. 1. Components of the cryptocurrency architecture covered in this article



coins of the user. These wallets can reside on a local client machine or be managed by an online service provider. In these applications, authenticating users and maintaining confidentiality of cryptographic key material are the central security goals. Exchanges facilitate trade between cryptocurrencies and between cryptocurrencies and traditional forms of money. Wallets broadcast cryptocurrency transactions to the network, whose participants relay transactions to miners, who in turn validate and group them together into blocks that are appended to the blockchain.

Not all cryptocurrency applications revolve around payments. Some cryptocurrencies, most notably Ethereum, support "smart contracts" in which general-purpose code can be executed with integrity assurances and recorded on the distributed ledger. An explosion of token systems has appeared, in which particular functionality is expressed and run on top of a cryptocurrency [12]. Here, the promise is that business logic can be specified in the smart contract and confidently executed in a distributed fashion.

The emergence of a vibrant ecosystem of decentralized cryptocurrencies has prompted proposals that leverage the underlying technology to construct new central bank currency [2] and corporate electronic money, such as Facebook's asset-linked Libra. This article focuses on existing decentralized cryptocurrencies. Some lessons discussed here could also inform the design and operation of these prospective forms of digital money issued by public or private legal entities.

Bugs in cryptocurrencies. The cryptocurrency realm itself is a virtual "wild west", giving rise to myriad protocols each facing a high risk of bugs. Projects rely on complex distributed systems with deep cryptographic tools, often adopting protocols from the research frontier that have not been widely vetted. They are developed by individuals with varying level of competence (from enthusiastic amateurs to credentialed experts), some of whom have not developed or managed production-quality software before. Fierce competition between projects and companies in this area spurs rapid development, which often pushes developers to skip important steps necessary to secure their codebase. Applications are complex as they require the interaction between multiple software components (e.g., wallets, exchanges, mining pools). The high prevalence of bugs is exacerbated by them being so readily monetizable. With market capitalizations often measured in the billions of dollars, exploits that steal coins are simultaneously lucrative to cybercriminals and

damaging to users and other stakeholders. Another dimension of importance in cryptocurrencies is the privacy of users, whose transaction data is potentially viewable on shared ledgers in the blockchain systems on which they transact. Some cryptocurrencies employ advanced cryptographic techniques to protect user privacy, but their added complexity often introduces new flaws that threaten such protections.

Disclosures. Disclosures in cryptocurrencies have occurred in varying circumstances, from accidental discoveries, through analysis by expert developers and academics, to observing successful exploits in the wild. In the rest of the paper we highlight the difficulties and subtleties that arise in each case. The root causes of most of the difficulties lie in the special nature of cryptocurrencies: they are based on distributed systems that were designed to be hard to change in order to provide strong guarantees on their future behavior. In order to change these rules the consent of many participants is needed – participants who are often anonymous, and who are organized loosely in communities without governing bodies or regulatory oversight.

We proceed as follows: after briefly highlighting the differences between conventional software development and cryptocurrencies with regards to vulnerability disclosure, we identify key issues in the disclosure process for cryptocurrency systems. Finally, we formulate recommendations and pose open questions.

2 HOW IS DISCLOSURE DIFFERENT?

Responsible vulnerability disclosure in cryptocurrencies differs from the conventions adopted for general software products in several ways. Two fundamental differences arise from the very nature of cryptocurrencies.

First, the decentralized nature of cryptocurrencies, which must continuously reach system-wide consensus on a single history of valid transactions, demands coordination among a large majority of the ecosystem. While an individual can unilaterally decide whether and how to apply patches to her client software, the safe activation of a patch that changes the rules for validating transactions requires the participation of a large majority of system clients. Absent coordination, users who apply patches risk having their transactions ignored by the unpatched majority.

Consequently, design decisions such as which protocol to implement or how to fix a vulnerability must get support from most stakeholders to take effect. Yet no developer or maintainer naturally holds the role of coordinating bug fixing, let alone commands the authority to roll out updates against the will of other participants. Instead, loosely defined groups of maintainers usually assume this role informally.

This coordination challenge is aggravated by the fact that unlike "creative" competition often observed in the open source community (e.g., Emacs versus vi), competition between cryptocurrency projects is often hostile. Presumably, this can be explained by the direct and measurable connection to the supporters' financial wealth and the often minor technical differences between coins. The latter is a result of widespread code reuse [28], which puts disclosers into the delicate position of deciding which among many competing projects to inform responsibly. Due to the lack of formally defined roles and responsibilities, it is moreover often hard to identify who to notify within each project. Furthermore, even once a disclosure is made, one cannot assume the receiving side will act responsibly: information about vulnerabilities has reportedly been used to attack competing projects [18], influence investors, and can even be used by maintainers against their own users.

The second fundamental difference emerges from the widespread design goal of "code is law", i.e., making code the final authority over the shared system state in order to avoid (presumably fallible) human intervention. To proponents, this approach should eliminate ambiguity about intention, but it inherently assumes bug-free code. When bugs are inevitably found, fixing them (or not) almost guarantees at least someone will be unhappy with the resolution. This is

perhaps best exemplified by the controversy around the DAO, an Ethereum smart contract with a reentrance bug that was exploited to steal coins worth around $50 million. After a community vote, the Ethereum developers rolled out a patch to reverse the heist, which (maybe surprisingly) turned out to be controversial. While the patch was accepted by large parts of the ecosystem, it was strongly opposed by a minority of Ethereum users arguing that it is a direct violation of the code-is-law principle, and the controversy ultimately led to a split of the Ethereum system into two distinct cryptocurrencies Ethereum and Ethereum Classic [1]. Moreover, situations may arise where it is impossible to fix a bug without losing system state, possibly resulting in the loss of users' account balances and consequently their coins. For example, if a weakness is discovered that allows anybody to efficiently compute private keys from data published on the blockchain [16], recovery becomes a race to move to new keys because the system can no longer tell authorized users and attackers apart. This is a particularly harmful consequence of building a system on cryptography without any safety net. The safer approach, taken by most commercial applications of cryptography but rejected in cryptocurrencies, places a third party in charge of resetting credentials or suspending the use of known weak credentials.

Ironically, these fundamental differences stem from design decisions intended to enhance security. Decentralization is prized for eliminating single points of control, which could turn out to be single points of failure. Giving code the final say is intended to preserve the integrity of operations. However, what may benefit security at design time becomes a significant liability after deployment once vulnerabilities are found.

Besides these fundamental differences, responsible disclosure for cryptocurrencies is characterized by specific features of the domain. The interpretation of system state as money, with many exchanges linking it mechanically to the conventional financial system, makes it easier and faster to monetize bugs than for conventional software, where vulnerability markets may exist but are known to be friction-prone [23]. Moreover, the cryptocurrency ecosystem reflects conflicting worldviews, which prevent the establishment of basic norms of acceptable behavior. For example, invalidating ransomware payments via blacklisting has reignited the debate over censorship versus the rule of law [26].

Finally, we note a difference in emphasis over certain aspects of disclosure. The conventional responsible disclosure discussion has focused on balancing users' interests in defensively patching versus national security interests of weaponizing vulnerabilities [25, 31], without regard to whether the affected software is open or closed source. By contrast, open-source software and code reuse are central to disclosure issues in the cryptocurrencies, whereas balancing national and individual security considerations has so far not been widely discussed.

Throughout the rest of the article, we illustrate these differences with real cases before we derive recommendations and point to open problems.

3 CASE STUDIES

We now review selected case studies of cryptocurrency vulnerability disclosures, highlighting aspects that teach us about the difficulties in response. We employ a multi-perspective method in selecting and researching these cases, ranging from the authors' direct experience as disclosers, interviews with developers and cryptocurrency designers, and through public reports. Interviews with open-ended questions were conducted by telephone, in-person or by email. Attribution is given unless the subject requested anonymity. The novelty and heterogeneity of the problem precluded a more systematic approach, though we hope that those informed by our findings can do so in future investigations. We investigate coins both small and large, because even the top coins have experienced severe bugs. While the software development processes for prominent coins are more robust, the cases will show that all coins experience challenges to disclosure not seen in traditional software projects. Fig. 2 presents a stylized timeline of the cases presented.

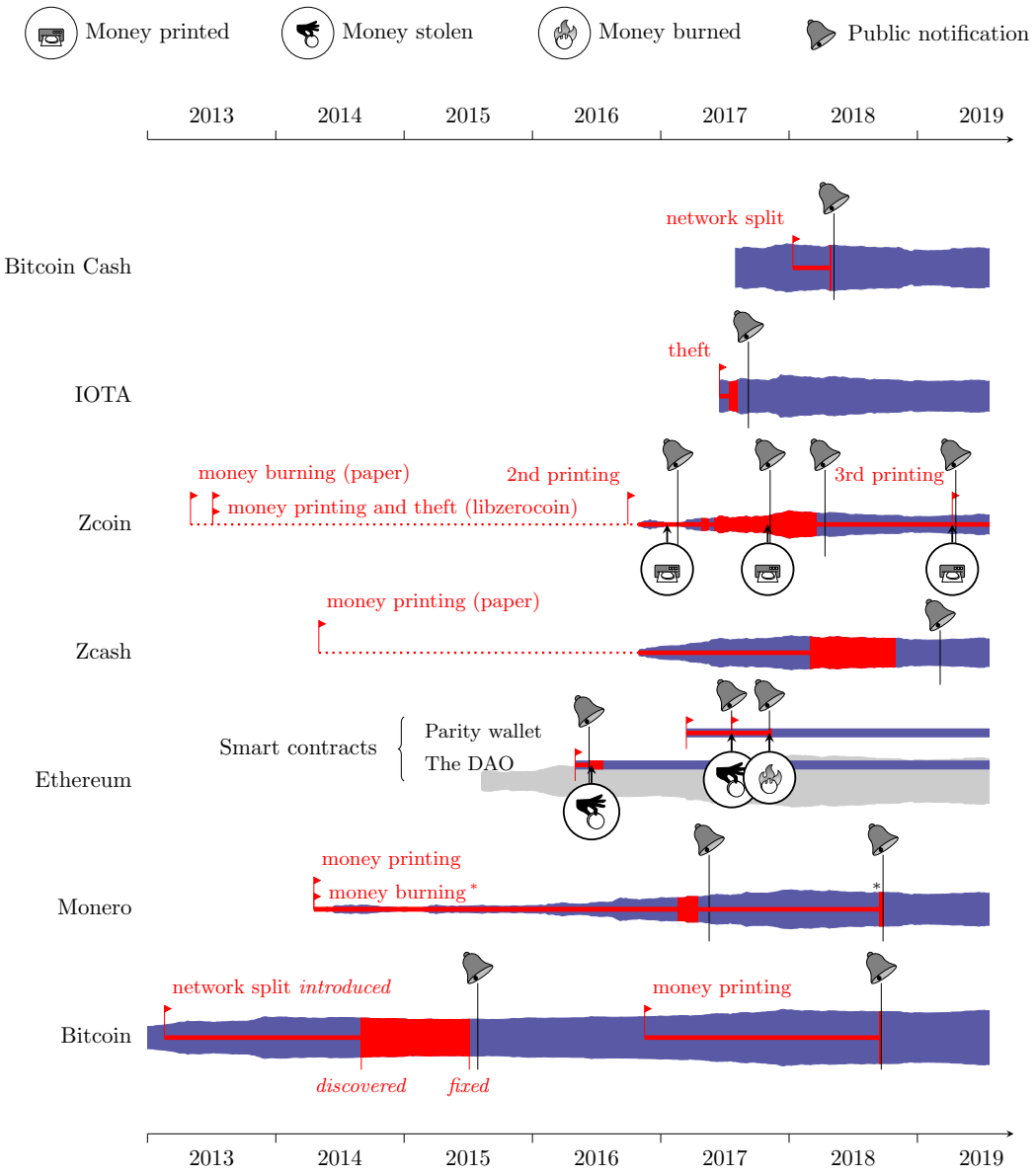


Fig. 2. Visualization of the vulnerabilities discussed in this article. The blue bars represent the underlying coins and their widths are proportional to their marketcap (c.f. Coinmarketcap.org). The red bars visualize the discussed incidents from their introduction (flag) to their disclosure (wide bar) to their public announcement (bell). The additional symbol is used whenever money was stolen, burnt or printed.

3.1 Cryptocurrency Systems

Zcoin. We start with Zcoin, a relatively unpopular cryptocurrency that has suffered from repeated disclosures. Zcoin was the first to implement the Zerocoin protocol [22], which uses zero-knowledge proofs to enable untraceable transactions. In February 2017, an attacker exploited a typo in C++ code (using the equality operator `'=='` instead of the assignment operator `'='` [17]) to generate 403,050 coins out of thin air. The new coins had a market value of $750,000 and inflated the currency supply in circulation by 37%. In principle, such attacks can remain unnoticed due to the zero-knowledge veil, but the sheer number of coins created combined with the attacker's impatience eventually led to its discovery. Within hours, the Zcoin team demanded that trading halt at big exchanges, published a blog post, and asked mining pools to suspend processing zero-knowledge transactions. A patch was released within a day, but the zero-knowledge feature remained disabled, thereby temporarily freezing all untraceable funds. This issue was resolved after four days when a "fork" altering the fundamental transaction validation rules was adopted by a majority of the miners. Even so, the attacker could still abscond with the heist.

Later in 2017, a team of researchers including author Ruffing found another vulnerability in Zcoin that allowed an attacker to "burn" money in transit, i.e., ensure that no one, including the sender, recipient, and attacker, can further spend the coins [30]. Remarkably, the root cause of this vulnerability was an overlooked attack vector in the design and security analysis of the underlying Zerocoin protocol. While money burning does not serve the attacker directly, the attacker could profit indirectly, e.g., by betting on falling prices of the affected cryptocurrency (short selling) and then publishing or exploiting the vulnerability. We have no evidence that such short-selling activity did indeed take place.

Having no cryptographer on its team, Zcoin hired Ruffing to provide advice and develop a patch. During the work, he identified two more vulnerabilities [29], one enabling illegitimate coin generation and one allowing theft of money in transit. Both vulnerabilities stemmed from bugs in `libzerocoin`, a prototype library written by the inventors of the Zerocoin protocol for the purpose of validating their research. The Zcoin project had used that library as-is, despite the code's prominent warning that the authors "are releasing this dev version for the community to examine, test and (probably) break" and that there are things that they have "inevitably done wrong" [21].

Code reuse complicated the disclosure process of the three vulnerabilities [29]. Months after the initial notification, the discoverers found that more than 1,600 public GitHub repositories included verbatim copies of `libzerocoin`. Responsible and confidential disclosure to so many recipients is infeasible. Instead, the discoverers narrowed down the recipient set to less than ten actual cryptocurrency projects, four of which they deemed trustworthy enough to be informed additionally. None of the projects had a clearly defined contact point or process for handling vulnerabilities.

Competition between projects prevented a coordinated response. For example, the notified project did not reveal to the reporters which of their competitors were also vulnerable. Coordination is essential because the first project to patch reveals the vulnerability, leaving the others unprotected. One currency was actually exploited in this way, and ironically, Zcoin itself was targeted because the patch was not adopted quickly enough. Dealing with the entire situation required tact and judgment by the discoverers, and the potential for every mistake to be catastrophic furthers the discoverers' burden.

As a result of the coin creation bugs, Zcoin improved continuous monitoring of aggregated balances, which led to the discovery of another creation bug in April 2019. The project repeated the notification process described above, disabled the zero-knowledge features via an emergency fork, and informed three potentially affected competitors. It took ten days of investigation before a project developer identified the root cause in the design of the Zerocoin protocol. Unlike a simple implementation bug, there was no obvious way to fix the problem. The project's response was to migrate to

an entirely different zero-knowledge protocol, suspending untraceable transactions in the meantime and freezing the affected funds until the new protocol was deployed in July 2019 [37].

Zcash. Zcash, the commercial implementation of the Zerocash protocol [3], improves on Zerocoin's model for untraceable transactions. It too has suffered from similar issues [32]. The proposal for the used algorithm for generating cryptographic material allowed a parameter to be published that should have remained secret. (Incidentally, a security proof was omitted because the scheme was similar to a previous one known to be secure.) The published value could have been used to undetectably generate coins out of thin air. The problem was discovered internally in March 2018 and fixed after 240 days in conjunction with a scheduled upgrade of the zero-knowledge protocol. Before and during the events the Zcash team had entered mutual disclosure agreements with the two largest competitors who reuse Zcash code. These competitors were notified two weeks after the fix with a schedule for public disclosure within a maximum of 90 days, which then took place in February 2019, almost one year after the discovery [32]. Obscurity played a key role in this event: not only was the fix hidden in a larger update, the critical parameter was also removed from websites and a cover story spun around the "loss" of this piece of information. The intention of this obscurity was to protect Zcash's own interests and its users, as well as those of competing cryptocurrencies. On the downside, such long periods of obscurity may cast doubt on the trustworthiness of security claims in the future, and it remains unclear whether and to what extent the bug has been exploited.

Monero. The opposite of internal discovery is accidental public disclosure. This happened to Monero, the most popular implementation of the CryptoNote protocol [35]. In September 2018, an interested user posted a seemingly innocuous question to an online forum: "what happens if somebody uses a one-time account twice?" (paraphrased by the authors) [7]. Surprisingly, there was no protection against this action in the protocol. The revealed vulnerability allowed attackers to burn other people's funds. The problem was fixed within ten days without known incidents and publicly announced thereafter.

A more serious vulnerability in the CryptoNote protocol affected all cryptocurrencies based on it. A post on a specialized cryptography mailing list in February 2017 revealed an issue which implied a coin generation vulnerability in CryptoNote's basic cryptographic scheme [20]. The Monero team took note and developed a patch within 3 days and shared it privately with preferred parties, such as mining pools and exchanges. The true purpose of the patch was disguised in order to protect the rest of the users who were running vulnerable clients. After a fork to the validation rules that completely resolved the issue in Monero in April 2017, the Monero team informed other CryptoNote coins privately. One such coin, Bytecoin, was exploited immediately afterwards, resulting in the illegitimate generation of 693 million coins [18]. In a public disclosure that took place 15 days later, the Monero team described the aforementioned process and named unpatched competitors, including Bytecoin [20] (though Bytecoin claims that a patch had been issued to miners immediately after the exploit [18]). Perversely, the public disclosure attracted other investors to bid up the Bytecoin price. Its market capitalization grew five-fold, briefly jumping into the top 10 cryptocurrencies by value. It remains unclear who exploited the bug, but Bytecoin holders certainly benefited from the price rise.

IOTA. Unlike bugs in which coins are created, IOTA suffered a vulnerability that might have placed user funds at risk of theft. Contrary to the best practice of using standardized cryptographic primitives, IOTA relied on a custom hash function that had a collision weakness [14]. Author Narula and colleagues disclosed the vulnerability to the developers in July 2017. The vulnerability was patched by IOTA in August 2017 and made public by the disclosers in September 2017 [13], offering several lessons about the disclosure process.

First, the vulnerability was fixed and deployed to the network quite quickly. On one hand, this is good because the potential vulnerability window is smaller. On the other hand, the speedy response was made possible due to the project's high level of control over the network, which runs contrary to the design goals of decentralized cryptocurrencies. Such control further allowed the operators to shut down its network to prevent theft from a vulnerable wallet for several weeks in early 2020.

The second lesson is that organizations may not respond favorably to a disclosure. Here, communications were tense, the existence and risk of the vulnerability was denied and downplayed, and the discoverers were threatened with lawsuits. The response echoes industry reactions to vulnerability disclosures related to digital rights management decades before [19]. In the cryptocurrency case, there is a clear potential incentive conflict when the organization holds a large share of the coins and reasonably worries that the news could devalue holdings or prevent partnerships that might increase the value of holdings. Moreover, information about the bug could be exploited for profit by those possessing inside information about its existence prior to public disclosure.

Bitcoin Cash. Not to be confused with Bitcoin, "Bitcoin Cash" is derived from Bitcoin's codebase and was created due to disagreements within the ecosystem. Cory Fields, a contributor to the predominant implementation of Bitcoin, Bitcoin Core, was examining change-logs of Bitcoin Cash's main implementation in April 2018 [10]. There he noticed that a sensitive piece of code dealing with transaction validation had been improperly refactored, causing a vulnerability. It would allow an attacker to split the Bitcoin Cash network, thereby compromising the consistency required for a cryptocurrency to operate.

As Fields noted, bugs like this cause systemic risk: if exploited, they could sink a cryptocurrency. The large amounts of money at risk prompt disclosers take precautions. In this case, to protect his own safety, Fields chose to remain anonymous [10]. The patching went smoothly, but we do not know if it would have been more contentious had he revealed his identity. Moreover, discoverers may want to demonstrate that they behaved ethically, for example, that they sent a report to the developers. One possible mechanism is to encrypt the report with the developers' public key and publish the ciphertext and draw the developer's attention to it. This would require developers to provide public keys along with their security contact and have internal processes to handle incoming messages. Surprisingly, at the time Bitcoin Cash, a top-10 cryptocurrency worth billions of dollars, did not (though now they do). In our interview, Fields stressed that he found it difficult to figure out what was the right thing to do. What helped him was to imagine the situation with swapped roles.

Bitcoin. A few months later, a developer from Bitcoin Cash disclosed a bug to Bitcoin (and other projects) anonymously. Prior to the Bitcoin Cash schism, an efficiency optimization in the Bitcoin codebase mistakenly dropped a necessary check. There were actually two issues: a denial-of-service bug and potential money creation [8]. It was propagated into numerous cryptocurrencies and resided there for almost two years, but was never exploited in Bitcoin.

This case teaches us three lessons. First, even the most watched cryptocurrencies are not exempt from critical bugs. Second, not all cases should be communicated to everyone in the network at the same time. The Bitcoin developers notified the miners controlling the majority of Bitcoin's hashrate of the denial-of-service bug first, making sure they had upgraded so that neither bug could be exploited before making the disclosure public on the bitcoin-dev mailing list. They did not notify anyone of the inflation bug until the network had been upgraded. Third, the disclosure involved deliberate deception of users: the Bitcoin developers published a patch describing it as only fixing the denial-of-service issue. This downplayed the severity of the bug, while at the same time motivating a prompt upgrade. This gave Bitcoin

users and other affected cryptocurrencies time to adopt the fix, albeit with grumbling about the sudden public release. This highlights both a benefit and a downside to employing white lies in the disclosure process.

Silence is an alternative to white lies. The Bitcoin team took this option after an internal discovery in 2014. Bitcoin suffered from an inconsistency between different versions of the `OpenSSL` library. The 32-bit version was more tolerant in accepting variants of digital signatures than the 64-bit version, which could cause a loss of consistency if a signature is accepted only by the subset of nodes running on 32-bit. The mitigation turned into a year-long ordeal. Fixing `OpenSSL` was not an option, hence the stricter signature format had to be enforced in the Bitcoin codebase. Changes were made subtly and gradually in order to avoid drawing attention on the relevant piece of code. Users upgraded organically over a period of 10 months. The bug was made public when more than 95% of the miners had patched [36].

3.2 Smart Contracts

Some cryptocurrencies, most prominently Ethereum, support "smart contracts". These are computer programs anyone can store on a shared blockchain, which then guarantees correct execution. Contracts can receive, store and send coins to users or other contracts according to their programmed logic. Smart contracts pose two further challenges to disclosure and patching. First, there is no club of miners whose incentives are aligned with the functioning of a specific contract. Therefore, relying on miners as allies to support smooth disclosure is usually not an option (though see below for an exception). Second, the code is not updateable by design to demonstrate commitment to the rules of operation, hence the contract analogy. This may turn disastrous if the code contains bugs because machines, unlike arbitrators of real contracts, have no room for interpretation.

The DAO. The most famous example of a buggy contract is the DAO (short for Decentralized Autonomous Organization), the first code-controlled venture fund. Widely endorsed by an enthusiastic Ethereum community, in spring 2016 the DAO project collected user funds and stored them in a smart contract. Its visible balance of $250 million (15% of all available coins at the time) made it a highly attractive target. It prompted scrutiny from security researchers who raised concerns [9], the closest activity to disclosure in the smart contract space we have seen. Three weeks later, an anonymous attacker managed to withdraw more than 3.5 million coins (about $50 million) illegitimately from the DAO smart contract [1]. The attacker's trick involved making a small investment in the DAO, then withdrawing and thereby exploiting a re-entrance vulnerability in the refund mechanism. (The contract's bug was to not decrease the balance before sending coins, which in Ethereum passes control to the receiving party.) This exploit set off a vigorous debate over whether or not this behavior was abusive, since the code technically allowed the interaction.

The DAO incident could have been an example of an irreversible change of system state. However, the exceptional scale of the project and the involvement of the Ethereum community triggered a historic vote between miners to support a fork of the underlying cryptocurrency in order to "restore" the investments in the DAO contract. This intervention was highly controversial as it thwarted the very idea of immutable transactions, causing a group of purists to create a parallel instance, called Ethereum Classic, that was not rolled back. In hindsight, the incident raised the alarm to the smart contract community about the looming security issues. Today's contracts cannot hope for miner-enforced rollbacks because the uptake of the platform has diversified interests.

Parity wallet. Another example of a fund recovery, albeit partially successful, followed the Parity exploit in July 2017. The vulnerable contract implemented a multi-signature wallet, a mechanism that promises superior protection against theft compared to standard wallets. Intended uses include "corporate" accounts storing high value, such as the proceeds from initial coin offerings (ICOs). An anonymous attacker observed a discrepancy between the published and reviewed

Table 1. Synthesis of recommendations

Dos	
	Provide point of contact including public key
	Liaise with competitors who share code
Don'ts	
	Single out vulnerable competitors
	Bug bounties in your own coin
Depends	
	Use obscurity and white lies during disclosure
	Notify all affected projects unless there is conflict
	Built-in notification and feature "kill" switches
Need for action	
	Clarify right or obligation to preventively move vulnerable funds
	Establish clearinghouse and coordinator

source code and the binary code, which was deployed for each of 573 wallets and omitted an essential access control step. This enabled a theft of coins worth $30 million from three accounts. Parity discovered the attack as it was ongoing and published an alert. This would have enabled attentive users to rescue their funds (exploiting the same vulnerability) in a race against the attacker and imitators. At this point, a total of another $150 million was essentially free to be picked up by anyone [33]. As expected, many users reacted slowly and found their funds missing. It turned out that a group of civic-minded individuals has taken the funds in custody in order to protect users and return them in a safe way. This example raises the question if protective appropriation of funds is legal, or should even be expected from discoverers.

Users who nevertheless continued to trust the Parity wallet software were less lucky following a second incident. The Ethereum platform has a fuse mechanism that irrevocably disables code at a given address. In November 2017, a user (allegedly) inadvertently invoked this mechanism on a library referenced in 584 intentionally non-updatable contracts of the next-generation Parity wallet. A total of $152 million was burned [34]. This time, no one intervened, presumably because the loss concerned only 0.5% of all coins.

We close by noting that as of this writing, we are not aware of any major cases of responsible disclosures of vulnerabilities in smart contracts.

4 RECOMMENDATIONS AND OPEN QUESTIONS

While best practices in secure software engineering and responsible disclosure [15] are increasingly adopted in the cryptocurrency space, there always remains a residual risk of damaging vulnerabilities. Therefore, norms and eventually laws for responsible disclosure need to emerge. What follows is a first step towards that end. Our synthesis of what can be learned from the cases is structured along three central issues of responsible disclosure: *(i)* how to protect users, *(ii)* who to contact, when and how, and *(iii)* how to reward the discoverer. Table 1 sums up the recommendations outlined in this section.

4.1 How to Protect Users

Discoverer safety. If the vulnerability can make parties who may operate beyond the law substantially richer or poorer, the discoverer's personal safety should be considered [10]. Death threats are not unheard of. Confidentially sharing the vulnerability with others the discoverer trusts (professional colleagues, notaries or the police) might reduce this risk. Sealed envelopes, or their digital variants such as time-locked encryption or secret sharing schemes, lessen the risk of unintended leakage. In addition, anonymous reporting may also reduce stress and tension. However, note that if the vulnerability is exploited, any proof that the discloser knew of the vulnerability before its exploit could be used as evidence the discloser was the attacker.

Addressing vulnerable funds. If a vulnerability means that anyone can steal money from an account, should civic-minded defenders proactively steal to protect funds, like in the Parity wallet case (Sect. 3.2)? This touches on unresolved legal questions. If "code is law" is the guiding principle, moving vulnerable funds must be legal. But courts are bound to real-world norms which differ across jurisdictions and circumstances. For example, in many places only law enforcement can legally expropriate property, including crypto coins. Elsewhere, disclosers could be *obligated* to intervene rather than stand by and allow a crime to take place. To give the discoverer legal certainty, it is essential to settle the basic question whether the discoverer could face legal consequences if she takes such precautions, or break the law if she has the power and does *not*. If opting to leave the matter to law enforcement, other complications arise: which law enforcement agency has jurisdiction and sufficient authority and is allowed to act? Do all law enforcement agencies possess the technical capability to intervene in time?

Preparing the system for disclosure. Given the inevitability of vulnerabilities, one strategy is to implement features in the cryptocurrency itself to automatically notify affected users of significant problems. In fact, Bitcoin used to have such an alert system, which enabled trusted actors to disseminate messages to all users and even suspend transactions. Such alert systems prompt difficult questions of their own, like who can be trusted with that authority in a decentralized system? Also, the alert system itself could become the target of attack, in much the same way that an Internet "kill switch" could create more security problems than it solves. Incidentally, Bitcoin itself abandoned the alert system over such concerns [4]. A similar idea is to incorporate a mechanism to turn off particular features if significant vulnerabilities are later found. Dash utilizes such a system that lets the holder of a secret key turn features on and off at will [27]. PIVX supports a similar mechanism to disable zero-knowledge transactions, which proved useful during the Zerocoin disasters (see Sect. 3.1).

Despite the benefits such features bring, they contradict the design philosophy of decentralization and might expose the privileged party to law enforcement requests. Supposing a cryptocurrency could overcome these challenges and develop mechanisms for disseminating protective instructions, the question of how to contact the trusted party who takes the precaution remains. We discuss this issue next.

4.2 Who to Contact, When, and How

Provide clear points of contact. Many cryptocurrencies are designed to avoid relying on privileged parties with substantial control. Yet this is in effect required to support responsible disclosure. It can be difficult to determine who is "in charge" (assuming anyone is) and who can fix the bug. Best practices recommend that developers provide clear points of contact for reporting security bugs, including long-term public keys [11]. Developers who reuse code are advised to publish the original contact information alongside their own to aid the search for affected projects.

Identifying the responder. All communication by the discoverer should serve the end of fixing the bug. This means that the discoverer needs to notify the party who is in the best position to solve the problem. For example, if the vulnerability affects the cryptocurrency's core implementation, then the developers are the natural responders. There is a long history of bugs in exchanges [24], in which case they would respond. It is important to note that once the responder has taken responsibility, the discoverer should adopt a "need-to-know" practice until the risk is mitigated. Sometimes the natural choice for responder is missing or untrustworthy. In this case, the discoverer can also serve as responder, or delegate the responsibility to a third party.

Responder communication with stakeholders. Given the decentralized nature of cryptocurrencies, the responder is usually not in a position to unilaterally act to fix the bug. Instead, the responder must seek stakeholders' support. This means communicating the right messages at the right time. It could be dangerous to tell the full truth right away, so the message may justifiably include obfuscation or even white lies. Different stakeholders might require varying levels of detail at particular points in time. For bugs that require certain transactions to be mined for successful exploitation, the responder might encourage miners to upgrade first in order to deploy a fix as fast as possible. Exchanges can suspend trading in order to limit price shocks as bad news breaks, or aid in blocking the transport of stolen funds. In other instances wallet developers need to be notified first, in order to deploy patches to their software. It is good practice to publish an advisory detailing the course of events and clarify any obfuscation or lies after the risk is mitigated. This transparency could mitigate the erosion of trust resulting from deception.

Coordination among multiple responders. As illustrated in the cases above, vulnerabilities often affect multiple projects. It is up to the discoverer to decide where to send the report. The reporter should be transparent about who has been informed. The discoverer can work with the responders to ensure that everyone affected has been notified. Coordination among responders is essential. Patches should be deployed as simultaneously as possible across affected projects, since the patching and publication of vulnerability information would leave others exposed if no precautions were taken. In some circumstances, the responders are competitors, and their attitudes towards one another range from suspicion to hostility. We discuss how to deal with such cases next.

Dealing with untrustworthy responders. While in the traditional security world, it is considered not only common courtesy but professionally and ethically required to inform other projects about vulnerabilities before disclosing their existence publicly. In the cryptocurrency world, one must adopt a more adversarial mindset. If the discoverer does not find a trustworthy responder, she can take on that responsibility. While one might not expect the discoverer to fix the bug, she could nonetheless take steps to protect users (see Section 4.1).

The situation is further complicated when multiple projects share a problem, and some are not trustworthy or are hostile towards each other. It is unreasonably burdensome for a discoverer to adjudicate such conflicts. Responders can make a best effort to identify affected parties (e.g., searching for coins sharing common codebases) and notify accordingly. This points to the need for developing a clearinghouse, à la CERT/CC.

External authorities. Banks, payment processors and other key financial institutions are often required to report vulnerabilities to banking regulators, who can coordinate the response if needed. There is no current equivalent for cryptocurrencies, and it is unclear under which jurisdiction such a thing would reside. Should some global reporting agency of this nature be formed? If so, how might it successfully operate given a community whose common ground is removing the need for central parties? An external body modeled on CERT/CC might serve as a useful starting point. A

less formal and more decentralized example to consider is iamthecalvary.org, an initiative bringing together security researchers with medical device manufaturers to promote responsible vulnerability disclosure and remediation.

4.3 How to Reward the Discoverer

The article has shown that disclosing a cryptocurrency vulnerability and reacting responsibly is very burdensome. Interviewees have reported sleepless nights and fears for their safety, which in turn has altered their professional collaborations and friendships. The alternative to profit from the vulnerability, potentially anonymously, is tempting. This is why cryptocurrencies specifically cannot expect altruistic behavior and must instead incentivize responsible disclosure [11].

Bug bounties offer an established way to reward those who find bugs [5]. It stands to reason that they would be a natural fit for cryptocurrencies, given that they have a built-in payment mechanism. However, denominating the reward in its own currency is problematic, since its value might diminish as a result of disclosing the vulnerability, and you are effectively rewarding the discloser in a currency which she just found to be buggy. Other approaches are possible – for example, Augur (a smart contract market platform) is experimenting with exploit derivatives. It is not unreasonable to think that the cryptocurrency community might innovate a solution that could be a model for the broader software community. Nevertheless, monetary rewards must complement and cannot substitute for healthy norms and a culture that welcomes vulnerability disclosure.

REFERENCES

[1] Nicola Atzei, Massimo Bartoletti, and Tiziana Cimoli. 2017. A Survey of Attacks on Ethereum Smart Contracts (SoK). In *Principles of Security and Trust (Proc. of POST'17) (Lecture Notes in Computer Science)*, Matteo Maffei and Mark Ryan (Eds.), Vol. 10204. Springer, 164–186.

[2] Morten Bech and Rodney Garratt. 2017. Central Bank Cryptocurrencies. *BIS Quarterly Review* 9 (2017), 55–70.

[3] Eli Ben-Sasson, Alessandro Chiesa, Christina Garman, Matthew Green, Ian Miers, and Madars Virza. 2014. Zerocash: Decentralized Anonymous Payments from Bitcoin. In *IEEE Symposium on Security and Privacy (S&P)*.

[4] Bryan Bishop. 2018. Alert Key Disclosure. Bitcoin development mailing list. https://lists.linuxfoundation.org/pipermail/bitcoin-dev/2018-July/016189.html.

[5] Rainer Böhme. 2006. A Comparison of Market Approaches to Software Vulnerability Disclosure. In *Emerging Trends in Information and Communication Security (ETRICS) (Lecture Notes in Computer Science)*, Günter Müller (Ed.), Vol. 3995. Springer, Berlin Heidelberg, 298–311.

[6] CoinMarketCap. 2019. Global Charts. https://coinmarketcap.com/charts/.

[7] dEBRUYNE. 2018. A Post Mortem of The Burning Bug. https://web.getmonero.org/2018/09/25/a-post-mortum-of-the-burning-bug.html.

[8] Bitcoin Core Developers. 2018. CVE-2018-17144 Full Disclosure. https://bitcoincore.org/en/2018/09/20/notice/.

[9] Mark Dino, Vlad Zamfir, and Emin Gün Sirer. 2016. A Call for a Temporary Moratorium on The DAO. Blog post. http://hackingdistributed.com/2016/05/27/dao-call-for-moratorium/.

[10] Cory Fields. 2018. Responsible disclosure in the era of cryptocurrencies. Blog post. https://medium.com/mit-media-lab-digital-currency-initiative/http-coryfields-com-cash-48a99b85aad4.

[11] Cory Fields and Neha Narula. 2018. Reducing the risk of catastrophic cryptocurrency bugs. Blog post. https://medium.com/mit-media-lab-digital-currency-initiative/reducing-the-risk-of-catastrophic-cryptocurrency-bugs-dcdd493c7569.

[12] Michael Fröwis, Andreas Fuchs, and Rainer Böhme. 2019. Detecting Token Systems on Ethereum. In *Financial Cryptography and Data Security (Proc. of FC'19)*, Ian Goldberg and Tyler Moore (Eds.).

[13] Ethan Heilman, Neha Narula, Thaddeus Dryja, and Madars Virza. 2017. IOTA Vulnerability Report: Cryptanalysis of the Curl Hash Function Enabling Practical Signature Forgery Attacks on the IOTA Cryptocurrency. https://github.com/mit-dci/tangled-curl/blob/master/vuln-iota.md.

[14] Ethan Heilman, Neha Narula, Garrett Tanzer, James Lovejoy, Michael Colavita, Madars Virza, and Tadge Dryja. [n. d.]. Cryptanalysis of Curl-P and Other Attacks on the IOTA Cryptocurrency. IACR Cryptology ePrint Archive, report 2019/344. https://eprint.iacr.org/2019/344.

[15] Allen D. Householder, Garret Wassermann, Art Manion, and Chris King. 2017. The CERT Guide to Coordinated Vulnerability Disclosure. Special Report CMU/SEI-2017-SR-022.

[16] Lee Hutchinson. 2018. All Android-created Bitcoin wallets vulnerable to theft. Ars Technica. https://arstechnica.com/information-technology/2013/08/all-android-created-bitcoin-wallets-vulnerable-to-theft/.

[17] Poramin Insom. 2017. Zcoin's Zerocoin Bug Explained in Detail. Blog post. https://zcoin.io/zcoins-zerocoin-bug-explained-in-detail/.

[18] A. M. Juarez. 2017. Fraudulent Transactions Allowed by the CryptoNote Key Image Bug Remain Valid. Archived version of Bytecoin GitHub issue http://archive.today/2017.05.24-094822/https://github.com/amjuarez/bytecoin/issues/104.

[19] Corie Lok. 2001. Dispute over Digital Music Muzzles Academic. *Nature* 411, 6833 (2001), 5.

[20] luigi1111 and Riccardo "fluffypony" Spagni. 2017. Disclosure of a Major Bug in CryptoNote Based Currencies. Blog post. https://web.getmonero.org/2017/05/17/disclosure-of-a-major-bug-in-cryptonote-based-currencies.html.

[21] Ian Miers. 2013. README of libzerocoin. https://github.com/Zerocoin/libzerocoin/blob/63ef50417d513194f7fd7294f8b44f6b5ae49f61/README.md#warning.

[22] Ian Miers, Christina Garman, Matthew Green, and Aviel D Rubin. 2013. Zerocoin: Anonymous Distributed E-Cash from Bitcoin. In *IEEE Symposium on Security and Privacy*.

[23] Charlie Miller. 2007. The Legitimate Vulnerability Market: Inside the Secretive World of 0-Day Exploit Sales. In *Workshop on the Economics of Information Security (WEIS)*. Carnegie Mellon University, Pittsburgh, PA.

[24] Tyler Moore, Nicolas Christin, and Janos Szurdi. 2018. Revisiting the Risks of Bitcoin Currency Exchange Closure. *ACM Transactions on Internet Technology* 18, 4 (2018), 50:1–50:18.

[25] Tyler Moore, Allan Friedman, and Ariel D. Procaccia. 2010. Would a 'cyber warrior' protect us: exploring trade-offs between attack and defense of information systems. In *New Security Paradigms Workshop (NSPW)*, Angelos D. Keromytis, Sean Peisert, Richard Ford, and Carrie Gates (Eds.). ACM, 85–94. https://tylermoore.utulsa.edu/nspw10.pdf

[26] Malte Möser and Arvind Narayanan. 2019. Effective Cryptocurrency Regulation Through Blacklisting. https://maltemoeser.de/paper/blacklisting-regulation.pdf.

[27] Dash project. 2017. Spork, Multi-Phased Fork. Glossary item in developer documentation. https://dash-docs.github.io/en/glossary/spork.

[28] Pierre Reibel, Haaroon Yousaf, and Sarah Meiklejohn. 2019. An Exploration of Code Diversity in the Cryptocurrency Landscape. In *Financial Cryptography and Data Security (Proc. of FC'19)*, Ian Goldberg and Tyler Moore (Eds.).

[29] Tim Ruffing, Sri Aravinda Krishnan Thyagarajan, Viktoria Ronge, and Dominique Schröder. 2018. A Cryptographic Flaw in Zerocoin (and Two Critical Coding Issues). Blog post. https://www.chaac.tf.fau.eu/2018/04/12/zerocoinzcoinpivxzoinsmartcashhexxcoin-attack/.

[30] Tim Ruffing, Sri Aravinda Krishnan Thyagarajan, Viktoria Ronge, and Dominique Schröder. 2018. (Short Paper) Burning Zerocoins for Fun and for Profit - A Cryptographic Denial-of-Spending Attack on the Zerocoin Protocol. In *Crypto Valley Conference on Blockchain Technology, CVCBT 2018, Zug, Switzerland, June 20-22, 2018*. IEEE, 116–119. https://doi.org/10.1109/CVCBT.2018.00023

[31] Ari Schwartz and Robert K. Knake. 2016. Government's Role in Vulnerability Disclosure. Harvard Kennedy School Discussion Paper.

[32] Josh Swihart, Benjamin Winston, and Sean Bowe. 2019. Zcash Counterfeiting Vulnerability Successfully Remediated. Blog post. https://electriccoin.co/blog/zcash-counterfeiting-vulnerability-successfully-remediated/.

[33] Parity Technologies. 2017. The Multi-sig Hack: A Postmortem. https://www.parity.io/the-multi-sig-hack-a-postmortem/.

[34] Parity Technologies. 2017. A Postmortem on the Parity Multi-Sig Library Self-Destruct. https://www.parity.io/a-postmortem-on-the-parity-multi-sig-library-self-destruct/.

[35] Nicolas van Saberhagen. 2013. Cryptonote v 2.0. White Paper. https://cryptonote.org/whitepaper.pdf.

[36] Pieter Wuille. 2015. Disclosure: Consensus Bug Indirectly Solved by BIP66. Bitcoin development mailing list. https://lists.linuxfoundation.org/pipermail/bitcoin-dev/2015-July/009697.html.

[37] Reuben Yap. 2019. Further Disclosure on Zerocoin vulnerability. Blog post. https://zcoin.io/further-disclosure-on-zerocoin-vulnerability/.

[38] Aviv Zohar. 2015. Bitcoin: Under the Hood. *Commun. ACM* 58, 9 (2015), 104–113.