

# ABC: A Cryptocurrency-Focused Threat Modeling Framework

Ghada Almashaqbeh<sup>1\*</sup>, Allison Bishop<sup>1,2\*\*</sup>, and Justin Cappos<sup>3</sup>

<sup>1</sup> Columbia University, NY, USA {ghada, allison}@cs.columbia.edu

<sup>2</sup> Proof Trading, NY, USA

<sup>3</sup> New York University, NY, USA jcappos@nyu.edu

**Abstract.** Cryptocurrencies are an emerging economic force, but there are concerns about their security. This is due, in part, to complex collusion cases and new threat vectors that could be missed by conventional security assessment strategies. To address these issues, we propose ABC, an **A**sset-**B**ased **C**ryptocurrency-focused threat modeling framework capable of identifying such risks. ABC’s key innovation is the use of collusion matrices. A collusion matrix forces a threat model to cover a large space of threat cases while simultaneously manages this process to prevent it from being overly complex. Moreover, ABC derives a system-specific threat categories that account for the financial aspects and the new asset types that cryptocurrencies introduce. We demonstrate that ABC is effective by conducting a user study and by presenting real-world use cases. The user study showed that around 71% of those who used ABC were able to identify financial security threats, as compared to only 13% of participants who used the popular framework STRIDE. The use cases further attest to the usefulness of ABC’s tools for both cryptocurrency-based systems, as well as a cloud native security technology. This shows the potential of ABC as an effective security assessment technique for various types of large-scale distributed systems.

## 1 Introduction

Cryptocurrencies and blockchain technologies are an emerging economic force. Since Bitcoin emerged in 2009 [63], the number of these “digital currencies” has grown into the thousands in 2019, with a total market capital exceeding \$380 billion [16]. As the value of these currencies has grown, their goals also changing. Early systems focused only on providing a virtual currency exchange medium [26, 63], but nowadays there is increasing interest in providing other types of distributed services, such as computation outsourcing [18] or file storage [36], on top of this medium. These newer applications suggest increased adoption of cryptocurrency-based systems in the coming years.

Yet, despite the many advantages they offer — decentralization, transparency, and lowered service costs — there is still a big gap between the promise of

---

\* Supported by NSF CCF-1423306.

\*\* Supported by NSF CCF-1423306 and NSF CNS-1552932.

cryptocurrencies and their performance in practice. A major stumbling block is the perception that these systems are not secure, and the large number of security breaches announced in the past few years give credence to these doubts [2, 4–7, 11–13, 15, 17, 19, 20, 23–25, 29, 31, 33, 34]. Therefore, a better understanding of the security of cryptocurrencies is needed in order to ensure their safe deployment in emerging applications, and their continued adoption.

The best practice for designing a secure system requires a threat modeling step to investigate potential security risks. Such a model can guide developers in deploying the proper countermeasures during the design phase, and in assessing the system’s security level in the after-design stage. Although threat modeling has been thoroughly studied in the literature, existing paradigms primarily target software applications [51] or distributed systems that have a small number of participant types [62]. These threat modeling techniques were not designed to be scalable for a set of diverse, mutually distrustful parties as is found in cryptocurrencies. Such systems, especially those providing distributed services, consist of parties that play different roles (miners, clients, and different types of servers), and an attacker may control any subset of these parties. Adding to the complexity, the attacker may seek to target any role in the system and may launch a diverse array of attacks with different intended outcomes. As these sets grow, the complexity of reasoning about and managing threat cases becomes unwieldy.

To address these issues, we propose ABC, an **A**sset-**B**ased **C**ryptocurrency-focused threat modeling framework. ABC introduces a novel technique, called a collusion matrix, that allows users to investigate the full threat space and manage its complexity. A collusion matrix is a comprehensive investigation and threat enumeration tool that directly addresses collusion by accounting for all possible sets of attacker and target parties. ABC reduces the combinatorial growth of these cases by ruling out irrelevant scenarios and merging threat cases that have the same effect. This explicit consideration of attacker collusion is particularly important in permissionless cryptocurrencies that allow anyone to join.

ABC’s models are also tailored to better consider the threat domain of cryptocurrencies. This is done by introducing new threat categories that account for the financial motivations of attackers and new asset types, i.e., critical components, these systems introduce. ABC identifies these categories by listing the assets in a system, such as the blockchain and the peer-to-peer network, and outlining what entails secure behavior for each one. Then, the threat categories are defined as any violation of the security requirements for these assets. This approach produces a series of system-specific threat classes as opposed to an a priori-fixed list of generalized ones. Another feature of ABC is acknowledging that financial incentives and economic analysis can play major roles in other steps in the design process. These tools can be used in risk assessment and in mitigating some types of attacks that cannot be neutralized cryptographically.

To demonstrate the framework’s effectiveness, we conducted a user study and prepared use cases. The study compared the performance of subjects in building threat models using ABC and the popular framework STRIDE. Among

the obtained results, we found that around 71% of those who applied ABC were able to identify financial threats in a cryptocurrency system, as compared to less than 13% of those applying STRIDE. In addition, while none of the STRIDE session participants spotted collusion between attackers, around 46% of those who used ABC identified these scenarios.

For the use cases, we applied ABC to four real-world systems: Bitcoin [63], Filecoin [36], CacheCash, and SPIFFE [30]. These cases attest to the usefulness of ABC’s tools, as they integrated well into CacheCash’s design phase, and revealed several missing threat scenarios in the public design of Filecoin. Furthermore, ABC proved to be advantageous in a collaboration between one of this paper’s authors and a team working on assessing and improving the security of SPIFFE, a cloud-based identity production framework hosted by the Linux Foundation. Using ABC, the SPIFFE group was able to reason about all threat cases in a systematized way, determine the critical threats, and then prioritize mitigation actions accordingly. This confirms the potential of ABC as an effective tool for assessing and improving security not only for cryptocurrency-based systems, but for large-scale distributed systems in general.

## 2 Related Work

To orient readers to the current state-of-the-art in threat modeling, we summarize here some prior work done in this area. We also highlight relevant works on threat identification and security analysis in cryptocurrencies.

**Threat modeling frameworks.** The STRIDE framework, developed by Microsoft as part of its Security Development Lifecycle (SDL), is one of the earliest and most popular works in this field [51, 69]. STRIDE is an acronym of the threat categories the framework covers, namely, Spoofing, Tampering, Repudiation, Information disclosure, Denial of service, and Elevation of privilege. This framework is a multistep procedure that involves understanding the software application functionality, capturing its operation flow using data flow diagrams (DFDs), mapping the components of these DFDs to the threat categories mentioned previously, and employing threat tree patterns to derive concrete threat cases.

Though several solutions have extended STRIDE to accommodate more complex systems [62], and cover other security requirements, e.g., privacy [45, 56], its model does not fit cryptocurrencies. Another study [71], in which the authors extended STRIDE’s threat categories to handle Bitcoin-like community currencies, bears out this premise. However, their approach targets only community fund operation; more modifications would be needed to handle other components of the currency exchange medium, and other types of distributed services.

Other paradigms have pursued slightly different approaches. KAOS [70] is a goal-oriented requirements engineering framework that has been extended to cover security. It analyzes the anti-security goals of a system to identify the types of threats that represent potential compromises. T-MAP [43] is a value-

driven framework that targets commercial off-the-shelf systems. It identifies all attack paths and assigns them severity weights based on the organization (or business) policy to help in evaluating security practices. ANOA [41] is a generic framework to define and analyze anonymity of communication networks, while the frameworks presented in [38, 50] target the secure design of data routing protocols. Finally, other works build specialized threat models for specific classes of distributed systems, e.g., storage systems [48], virtual directory services [44], and unmanned aerial vehicle systems [52], rather than introducing a framework.

The aforementioned works indicate that different types of systems have different requirements when performing threat modeling. This reinforces the idea that emerging systems, such as cryptocurrencies, need specialized threat modeling tools.

**Security analysis of cryptocurrencies.** Most of the work done so far in this category can be divided into two classes. The first class formalizes the security properties of consensus protocols and blockchains [46, 65], with the goal of providing a security notion and a rigorous framework to prove security of blockchain-based systems, whereas the second class discusses specific security attacks on cryptocurrencies. For example, in a series of studies on Bitcoin, Boneau et al. [42] present several security threats, Androulaki et al. [39] evaluate its anonymity property, Gervais et al. [47] study how tampering with the network links affects participants’ view of the blockchain, and Kroll et al. [55] study the economics of Bitcoin mining and the effect of miner financial incentives on its security.

Other works dealing with different types of cryptocurrencies include Luu et al. [57, 58] who focus on security threats to smart contracts in Ethereum [72], Kosba et al. [54] who provide a model for decentralized smart contracts that preserve user’s privacy, and Sanchez et al. [59, 60] who analyze the security of Ripple [26] and the linkability of wallets and transactions in its network. Moreover, a recent empirical study, by Moser et al. [61], shows that the transactions in the privacy preserving cryptocurrency Monero [22] are traceable and that their real inputs can be identified. And another, by Kappos et al. [53], shows that the anonymity set of the private transactions in Zcash [35] can be shrunk using a simple heuristic derived from coin usage patterns.

While these attack descriptions are very useful, they only outline specific threat scenarios for a given system. Our goal, however, is to develop a framework that allows reasoning about the full set of potential attacks facing any cryptocurrency-based system.

### 3 Stepping through the ABC Framework

Having highlighted the need for a cryptocurrency-specific threat modeling framework, we now present the ABC model that answers to this need. As a systematized approach, applying ABC starts by understanding the functionality of the cryptocurrency system under design with a focus on its asset types and the finan-

cial motivations of the participants (Section 3.1). This is followed by identifying the impactful threat categories and mapping them to the system assets (Section 3.2). After that, ABC directs system designers to extract concrete attack scenarios using a new tool called a collusion matrix, which helps in exploring and analyzing the full threat space (Section 3.3). Lastly, ABC acknowledges that financial incentives affect other design steps, including risk assessment and threat mitigation (Section 3.4).

To make the discussion easier to follow, we illustrate the ABC process by describing its application to the following simplified system, which was inspired by Golem [18]:

**CompuCoin** is a cryptocurrency that provides a distributed computation outsourcing service. Parties with excessive CPU power may join the system as servers to perform computations on demand for others. Clients submit computation jobs to servers, wait for the results and proofs of correctness, and then pay these servers with cryptocurrency tokens. The mining process in CompuCoin is tied to the amount of service provided to the system. That is, the probability of a server being selected to mine the next block on the blockchain is proportional to the amount of computation it has performed during a specific period.

The full threat model for CompuCoin is available online [32]. Several excerpts from this model are embedded in the discussion of ABC steps that follows.

### 3.1 System Model Characterization

Understanding the system is an essential step in the threat modeling process. A misleading or incomplete system description can lead a designer to overlook serious threats and/or incorporate irrelevant ones. Therefore, an accurate system model must outline the use scenarios of the system, the assumptions on which it relies, and any dependencies on external services. In addition, the model must be aware of all participant roles, and any possible motivation each might have to attack the system. For the latter, evaluators need to consider how the financial interests of these participants shape their behaviors.

Moreover, a system model must define the critical components that need to be protected from attackers. These components represent the assets that would compromise the whole system if attacked. To capture the features of the system, ABC identifies these assets based on functionality. In detail, ABC divides the system into modules, and labels the valuable components of each module as assets, which could be concrete or abstract resources [62]. For example, the blockchain and the currency can be considered concrete assets, while preserving user privacy would be an abstract asset.

Finally, a system model includes graphic illustrations of its work flow. For distributed systems, it is useful to draw network models [62] in which system modules are represented by graphs showing all participants and assets, and the interactions between them. These graphs are helpful when enumerating the concrete threat scenarios as we will see in Section 3.3.

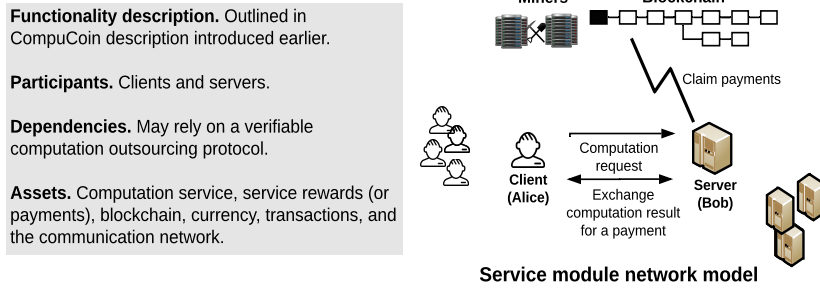


Fig. 1: System model characterization of CompuCoin.

**Running example application.** Figure 1 illustrates how this step would look in CompuCoin. It shows the various components of the system model, in addition to a network model of the computation outsourcing service. To cover the full functionality of the system, other network models would be needed to capture components, such as the mining and consensus processes. Furthermore, the asset list in this figure is not exhaustive, and is limited by the rather brief description provided for CompuCoin.

As shown in the figure, anyone can join CompuCoin as clients and servers, with servers also filling the role of miners. Dependency on other systems may include reliance on a verifiable outsourcing computation protocol, e.g. [64]. In terms of assets, one may define three in CompuCoin: the *Service* promised to clients, the *Payments* used to compensate for the service, and the *Currency Exchange Medium* that covers four sub-assets (in light of the extended review of Bitcoin [42]): the blockchain, currency, transactions, and the communication network that connects the parties together. Here one may merge the currency with the transactions in one asset, as transactions are usually the currency tokens that state the coin’s ownership. Another option is to merge currency with the payment asset to cover all currency flow in the system. However, we believe that a fine-grained division provides a more comprehensive treatment when identifying threats.

### 3.2 Threat Category Identification

After understanding the system model, the next step is to identify the broad threat categories that must be investigated. For each system component, system analysts outline all threat classes that may apply. Here ABC steps away from the conventional practice of using an a priori-fixed list, and instead uses an adaptive approach inspired by requirements engineering [45]. This approach defines threats as violations of system security goals. Given that assets are the target of security breaches, ABC defines these threat classes as violations of asset security requirements. This allows deriving system-specific threat categories because ABC identifies the assets in a way that aligns with the functionality of the system under design.

Table 1: CompuCoin threat categories.

Asset	Security Threat Category
Service	Service corruption (provide corrupted service for clients).
	Denial of service (make the service unavailable to legitimate users).
	Information disclosure (service content/related data are public).
	Repudiation (the server can deny a service it delivered).
Service payments	Service slacking (a server collects payments without performing all the promised work).
	Service theft (a client obtains correct service for a lower payment than the agreed upon amount).
Blockchain	Inconsistency (honest miners hold copies of the blockchain that may differ beyond the unconfirmed blocks).
	Invalid block adoption (the blockchain contains invalid blocks that do not follow the system specifications).
	Biased mining (a miner pretends to expend the needed resources for mining to be elected to extend the blockchain).
Transactions	Repudiation (an attacker denies issuing transactions).
	Tampering (an attacker manipulates the transactions in the system).
	Deanonymization (an attacker exploits transaction linkability and violates users' anonymity).
Currency	Currency theft (an attacker steals currency from others in the system).
Network	Denial of service (interrupt the operation of the underlying network).

Accordingly, in this step, an evaluator examines each asset and applies the following procedure to identify its threat classes:

- Define what constitutes secure behavior for the asset, and use that knowledge to derive its security requirements. These requirements include all conditions that, if met, would render the asset secure. For example, CompuCoin’s servers provide a computation outsourcing service and collect payments in return. One may consider the service payment asset secure if: a) servers are rewarded properly for their work, and b) that they earned the payments they collected.
- Define the threat categories of an asset as violations of its security requirements. Tying this to the above example, the service payment asset would have the following threat classes: service slacking, where a server collects payments without performing all the promised work, and service theft, where a client obtains service for a lower payment than the agreed upon amount.

The previous steps are highly dependent on how system analysts define the security properties of an asset, especially if there is no agreed-upon definition in the literature. For example, several works studied the security of the blockchain and the consensus protocol [42, 46, 65]. Yet, there is no unified security notion for the service asset because each type may have different requirements.

**Running example application.** Applying this step to CompuCoin produced the threat categories listed in Table 1 (the detailed process of deriving the security requirements and negating them to produce the listed threats for each of these assets is presented in Appendix A). We found this table useful when building threat models for all the use cases introduced in Section 5, where we mapped the listed categories to the assets in each system. In this mapping process, we found that some threat types were not applicable due to the absence of some assets. Notably, Bitcoin’s only assets are the ones related to the currency exchange medium. On the other hand, other systems required replicating some of these categories among all instances of an asset, e.g., in Filecoin all service asset threats were replicated for the two service types this system provides, namely, file storage and retrieval. This shows how the system characteristics affect the threat category identification step in ABC.

### 3.3 Threat Scenario Enumeration and Reduction

Once the threat categories have been identified, the next step is to enumerate concrete attack scenarios under each threat type. It is important in this step to be as comprehensive as possible by considering all potential attackers and target parties, as well as the set of actions attackers may follow, and the capabilities they must possess, to achieve their goals. This also involves considering collusion between several participants who may cooperate to attack the system.

Detecting collusion is particularly important in cryptocurrencies. The presence of monetary incentives may motivate attackers to collude in more ways than traditional distributed systems. The popular centralization problem caused by mining pools attests to this fact, as when these pools collude they can perform devastating attacks. Even miners may collude by accepting, or rejecting, updates on the network protocol which leads to hard forks in the system. ABC can detect these and other collusion cases at early stages of the system design.

To achieve this, ABC introduces collusion matrices that instruct analysts to enumerate all collusion cases, and reason about the feasibility of all threat scenarios in the system. A collusion matrix is two-dimensional, with the rows representing potential attackers and the columns representing the target parties. For the rows we list all participant roles in the system, both individually and in every possible combination. We also add a category called “external” that represents all entities outside the system. The same is done for the columns, with the exception that “external” is excluded. By definition, an external party is not part of the system, and hence, can not be a target. Each cell in these matrices represents a potential threat case to be investigated.

An example of a collusion matrix for the service theft threat in CompuCoin is shown in Figure 2. The dashed ellipse in the accompanying network model encloses a service session, which is an interaction between a server and a client. Any entry with multiple parties on the attacker side in this matrix indicates collusion. Note that a participant label may represent slightly different roles depending on where it is placed. For example, in Figure 2 the label “server” on the target side corresponds to a single server (i.e., Bob) since a service session



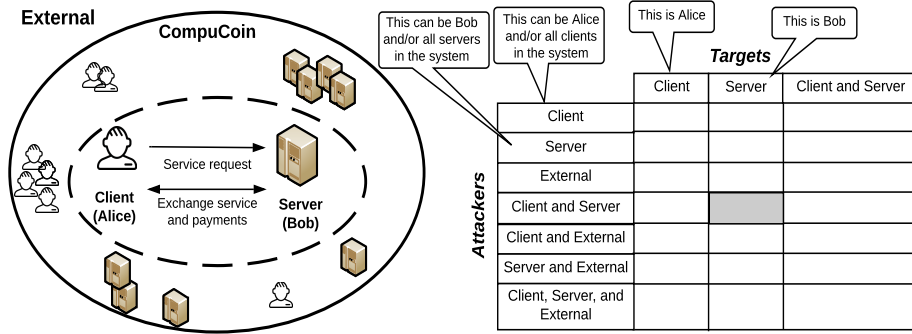


Fig. 2: Collusion matrix of service theft threat in CompuCoin.

involves only one server. However, the label “server” on the attacker side represents all servers in the system, including Bob. Hence, the cell in grey shade in Figure 2 does not suggest that a server colludes to attack itself, but instead, it represents the case where other servers collude with Alice against Bob.

For each threat category mapped to the assets in the system, a separate collusion matrix is created and analyzed as follows:

**1) Enumeration:** In this step, system analysts examine each cell and enumerate all strategies that attackers with specific capabilities can use against the target parties, and documenting the process. It is useful to consider the network model of the system components as they show the interactions between the participants and the system assets.

**2) Reduction:** While examining each cell, system analysts reduce the number of threat cases by:

- Eliminating cells representing scenarios that will not produce a threat to the system. This consists of crossing out the eliminated cells and documenting the rationale for elimination. For example, in Figure 2, the cells that have the client as a target are irrelevant to the service theft threat. This is because a client does not provide a service to others. Other cases can also be crossed out if they are neutralized by system assumptions or by early design choices. For example, requiring all transactions to be signed by their originators rules out transaction repudiation and tampering attacks.
- Merging together scenarios (and the corresponding cells) that have the same effect, or those that do not become stronger with collusion. For example, in Figure 2, the grey shaded cell in which Alice is colluding with other servers to avoid paying Bob is reduced to the case that Alice is a sole attacker. This is because only Alice pays for the service she receives from Bob, while other servers are not part of the protocol<sup>4</sup>.

<sup>4</sup> The case that these clients drop/withhold these payments in collusion with Alice is part of other threats, such as DoS attack.

Attacker	Target	Client	Server	Client and Server
External		Clients cannot be targets because they do not serve others.	Servers and external cannot attack because they do not ask/pay for service.	Reduced to the case of attacking servers only, clients do not serve others (cannot be targets).
Server				
Server and External				
Client			(1) Refuse to pay after receiving the service. (2) Issue invalid payments.	
Client and External			Reduced to the case of an attacker client. A client does not become stronger when colluding with other servers or external entities.	
Server and Client				
Client, Server, and External				

Fig. 3: Analyzing the collusion matrix of the service theft threat in CompuCoin.

**3) Documentation:** System analysts should document all threat scenarios that remain after the reduction step. That is, each documented case should outline the attack description, the target parties and assets, the attacker(s), the flow of actions, all preconditions that make the attack feasible, and the reasons behind merges and deletions (if any).

The overall number of matrices and the size of each matrix depend on the system parameters, such as number of participant roles and assets. The above reduction step eliminates a substantial number of cells in a documented, principled way, saving time and effort.

**Running example application.** The CompuCoin threat model has 11 collusion matrices [32]. We present one of them here: the service theft threat collusion matrix as illustrated in Figure 3. As shown, 21 cells can be reduced to just 2 threat scenarios (merged and ruled-out cells are displayed in pink and black shades, respectively). In this matrix, ten cases have been ruled out. These include all cells under the column with the “client” header, for the reasons explained previously, and the first three cells under the column with the “server” header. This is because “external” and/or “server” cannot be attackers because they do not ask/pay for the service<sup>5</sup>.

Ten other merged cases are shown in Figure 3. This includes all cells under the column with the “client and server” header, which are reduced to attacking only servers. This is again because clients do not serve others. The rest of the merges cover the last three cells in the column with the “Server” header. In these cells, a client is colluding with an external entity and/or other servers to make the target server lose payments. Such collusion will not make a client stronger as all these parties can do is drop/withhold payments, an option already covered under DoS threat. Hence, all these cells are reduced to the case of a solo client attacker.

<sup>5</sup> One may say that an external may join the system as a client to perform the attack. This case is covered under the client role in the matrix.

### 3.4 Risk Assessment and Threat Mitigation

The outcome of the threat modeling process, i.e., the documented list of impactful threat cases, gives the designers a guiding map to secure the system. During this process, it is useful to prioritize threats based on the amount of damage they can cause and the likelihood that an attacker has the required capabilities to carry out them. This falls under the purview of risk management, a separate task from threat modeling, carried out using frameworks like DREAD [51] or OCTAVE [37].

ABC integrates with risk management by leveraging existing techniques for threat mitigation. For example, many threat vectors can be addressed using rational financial incentives that are often called detect-and-punish mechanisms. That is, when a cheating incident is detected, the miners punish the attacker financially. Others can rely on designing algorithms that when executed in a malicious way consume more resources, i.e., incur a larger cost, than an honest execution. These approaches can use a game theoretic approach [68] to set the design parameters in a way that makes cheating unprofitable. By modeling interactions between the players as an economic game, the financial gain of all player strategies can be computed. Then, the parameters are configured to make honest behaviors more profitable than cheating.

The same procedure can be used to quantify the damage these financial threats may cause. In other words, a threat that could give the attacker a big payoff should be prioritized over a threat that yields minimal profits. This reinforces the idea that cryptocurrencies require an expanded model for exploring risks and countering them.

**Running example application.** To illustrate this step in CompuCoin, we consider the distilled threat scenarios found in Figure 3. Both threats can be neutralized financially by designing proper techniques to make the client lock the payments in an escrow and create a penalty deposit before asking for any service. The client loses the penalty deposit if it should cheat, perhaps by issuing invalid payments that carry its signature. The deposit amount needs to be at least equal to the maximum additional payoff a cheating client may obtain as compared to an honest client. This makes cheating unprofitable, and hence, unappealing to rational clients.

## 4 Evaluation

To evaluate the effectiveness of ABC, we set up an empirical experiment that compares how it performs against STRIDE [51], a widely used threat modeling framework. We chose STRIDE for this comparison because it is a popular example of the type of a model a system designer will turn to in the absence of a cryptocurrency-specific framework [71]. The experiment took the form of a user study in which participants were asked to build threat models for a simple cryptocurrency system using one of these two frameworks. Our primary goal was to test whether financial incentives and collusion could influence the type

of threats discovered. Thus, our evaluation focuses on answering the following questions:

1. Does a threat modeling framework affect how subjects characterize a system model?
2. Do the threat categories of each framework influence the broad threat classes identified by the subjects?
3. Do participants build more accurate threat models when using ABC than when using STRIDE?
4. Do participants find the ABC/STRIDE method easy to use in completing the study?

In what follows, we discuss the study methodology and some of the insights drawn from the findings.

#### 4.1 Methodology

We recruited 53 participants, primarily masters students in systems security programs. We used five subjects as a pilot group to test and refine our materials. The remaining 48 participants were divided randomly into two groups of 24, one of which built the threat model with STRIDE, whereas the other used ABC.

Each testing session spanned three hours and was divided into two parts: a group tutorial and individual completion of threat models. The group tutorial started with a 20 minute overview of cryptocurrencies, followed by a one-hour training in the framework to apply. The ABC tutorial contained a summary of the steps found in this paper, and for STRIDE, we prepared a tutorial based on material found in [21, 45, 51, 69]. The participants were then given a 25 minute break to reduce any fatigue effects. The session resumed with a 15 minute overview of ArchiveCoin, the system for which subjects will build a threat model. ArchiveCoin is a simplified Filecoin [36]-inspired cryptocurrency system that focuses mainly on the service and its rewards in order to fit the study session period.

During the remaining hour of the study session, individuals worked independently to complete a threat model. Given that the allocated time was short, we asked the subjects to look into just one threat category in Step 3, namely, the service theft of file retrieval. This category was not used in the clarifying examples of the tutorials to avoid biasing the results. Participants performed Steps 1 and 2 (system model characterization and threat category identification), and then submitted their answers. Only at this point were they given the materials for Step 3, in which they were asked to elicit threat scenarios for service theft of file retrieval. This was done so that participants who missed this threat when answering Step 2 could not alter their responses. At the end, the participants were asked to fill out a short questionnaire in which they rated how easy or hard it was to apply the threat modeling framework they employed. Our study instrument and all supporting materials are available online [32].

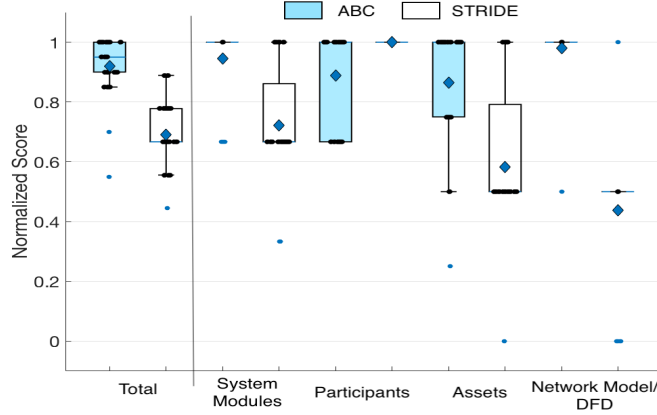


Fig. 4: Subject scores for Step 1. Diamonds indicate the mean.

## 4.2 Findings

In this section, we present the study results using the four questions outlined previously as a guideline.

**Effect on System Model Characterization** In the first step of each threat modeling framework, the subjects were asked to characterize the system model by defining its modules, its assets, and the participant roles, in addition to drawing either a network model of the system, in case of ABC, or a data flow diagram (DFD), in case of STRIDE. To quantify the influence of the framework on this step, we compute the subject scores using reference threat models we built for ArchiveCoin prior to the study<sup>6</sup>. We report these scores after normalization, meaning that we divide them by the maximum score value one may obtain when answering everything correctly.

The results for Step 1 are found in Figure 4<sup>7</sup>. As shown, ABC scored higher than STRIDE, with total average values of 0.92 and 0.69, respectively. Analyzing the responses for the sub-steps in Step 1 revealed several interesting observations. The first one is related to identifying the financial assets and modules in the system (depicted in Figure 5). As the figure shows, several subjects who applied STRIDE did not identify the payments (or currency) as an asset. Instead, their focus was on the user files stored in the system. Similarly, most of them did not

<sup>6</sup> We built two reference models, one using STRIDE and one using ABC to evaluate the responses of each framework session. Nonetheless, both models produced the same list of elicited threat cases in the last step.

<sup>7</sup> This figure is a box plot [14], which displays the distribution of the data points by showing the maximum and minimum values (the whiskers above and below the box), the median (horizontal line inside the box), and the data points that span the first to third quartiles (the box itself). In case most of these points are very close this box is suppressed into a line.



Fig. 5: Subject frequency of identifying payment related modules and assets.

identify the payment process as a system module, and focused only on file storage and retrieval processes. On the other hand, most of the subjects in the ABC session identified these financial related assets and modules. These results indicate that employing conventional threat modeling frameworks, instead of ones that are customized for monetary-incentivized systems, could lead evaluators to neglect the financial aspects of the system. This, in turn, could cause important threat cases to be overlooked, and thus, leave the system vulnerable to attacks.

The second observation is related to how subjects defined the participant roles in the system. As shown in Figure 4, STRIDE achieved an average score of 1 in this category as compared to 0.89 for ABC. All STRIDE session subjects defined the participant roles correctly, which in that model, included only clients and servers. For ABC, although its tutorial mentioned that an “external” entity must be considered among the participant roles, not all the subjects in that session recorded this role in their responses. This points to an important observation. Evaluators may only consider the insider attackers that interact with the system, and forget that external entities could be also motivated to, and capable of, an attack. Considering external attackers affects not only what concrete threat scenarios are elicited in Step 3, but also what threat categories are identified in Step 2. Therefore, more emphasis needs to be placed on this role early on in the threat modeling process.

Lastly, the third observation is related to how the framework influenced the way subjects represented the system modules graphically. Figure 4 shows that the average scores for the network model/DFD sub-step were found to be 0.98 and 0.44 for ABC and STRIDE, respectively. STRIDE session subjects struggled to draw a DFD for ArchiveCoin because such a representation is more suitable for software applications than distributed systems. On the other hand, ABC’s use of network models made this task easier for its session subjects, and almost all of them sketched diagrams correctly. As mentioned previously, this graphic representation helps in eliciting the concrete threat scenarios in the system (Step 3), and hence, inaccurate diagrams may affect the outcome of this process.

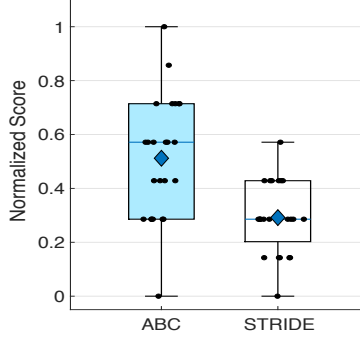


Fig. 6: ABC and STRIDE scores for Step 2. Diamonds indicate the mean.

**Effect on Threat Category Identification** In Step 2 of both frameworks, the subjects were asked to define the broad threat categories to be investigated. As part of the study material, participants who applied STRIDE were given its threat category list, along with the component mapping table found in the STRIDE user guide [21]. Similarly, participants who applied ABC were given the list found in Table 1 (covering only the service and service reward assets). The subjects in both groups defined the categories to be considered for ArchiveCoin by mapping these lists either to the system assets (in case of ABC), or to the DFD components (in case of STRIDE). The reference models we built indicated that the mapping outcome for both frameworks would include the following threat classes: service corruption, DoS, information disclosure, service slacking and theft for both service types that ArchiveCoin provides (file storage and retrieval).

Based on the scores for the threat identification step found in Figure 6, the cryptocurrency-tailored categories of ABC made it easier for the study participants to identify the threat categories in question as compared to STRIDE. This is despite the subjects having little experience with cryptocurrency-based systems. The average score for ABC subjects is around 0.51, compared to 0.29 for STRIDE (note these scores are normalized as mentioned before). The generalized categories used by STRIDE fit software applications well, but they do not suit monetary-incentivized distributed systems. System analysts, using these generalized categories, would need to expend more time and effort in order to identify the more specific threat classes of interest.

To provide more insights about this step, we analyzed the number of subjects who identified each threat category that need to be investigated. The results are depicted in Figure 7. We found that STRIDE’s subjects are ahead of ABC’s session participants for both DoS and information disclosure threats. As shown in Figure 7, around 88% and 83% of STRIDE subjects identified these categories, respectively, while around 50% and 42% of ABC’s subjects did so. Although we do not have a precise justification for this outcome, we think that this can be attributed to the threat category table of STRIDE, which thoroughly explains

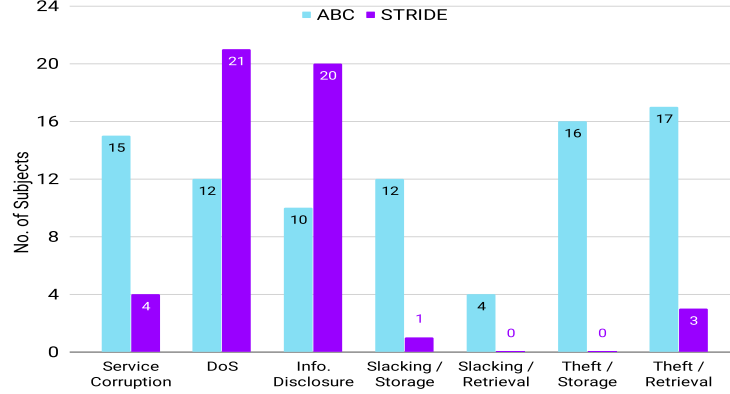


Fig. 7: Subject frequency of threat category identification.

these categories and provides detailed attack examples. Hence, we believe that the ABC tutorial needs to stress these threats and explain them in greater depth.

In contrast, ABC is ahead of STRIDE for all financial-related threats, i.e., service slacking and theft, as well as the service corruption threat. For the service theft of file retrieval, which is the category that we asked the participants to investigate in Step 3, only three participants in the STRIDE session spotted this threat, while 17 subjects in the ABC session did so, or around 13% and 71%, respectively. Furthermore, none of STRIDE participants spotted the service theft of file storage and slacking of file retrieval, while only one participant spotted service slacking of file storage. On the other hand, 67%, 17%, and 50% of ABC participants identified these categories, respectively. This, again, shows that the ABC threat classes guided the subjects toward service and payment related threats in a better way than the general categories included by STRIDE.

**Threat Model Accuracy** To quantify accuracy, we compute the recall and precision values for the concrete threat scenarios found by each subject as compared to the reference threat models we built for ArchiveCoin. The recall is computed as  $TP/(TP + FN)$ , and precision is computed as  $TP/(TP + FP)$ , where a true positive  $TP$  is a correctly identified threat, false negative  $FN$  is an undetected threat, and false positive  $FP$  is an incorrectly defined threat. The recall (precision) indicates how many valid (invalid) threats a subject defined. Both quantities take values between 0 and 1.

Based on the results of Step 3 in each framework (i.e., eliciting concrete threat scenarios), we found that participants who applied ABC produced a larger number of valid threat cases than the STRIDE session subjects, with average recall values of 0.48 and 0.4, respectively. At the same time, participants using ABC identified a lower number of irrelevant cases than those who applied STRIDE.



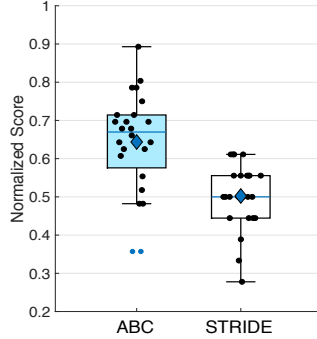


Fig. 8: Total normalized scores (diamonds indicate the mean).

The former scored an average precision value of 0.57, as opposed to 0.48 for the latter<sup>8</sup>.

We believe that the result above can be attributed to several factors. First, ABC directed participants to consider the financial aspects of the system, which affected the elicited threat scenarios. Second, the use of the collusion matrices helped ABC participants reason about the threat space in an organized way that reduced random speculations, as opposed to the STRIDE threat tree patterns that work well when applied to software applications. Third, ABC’s collusion matrices guided participants to spot threat cases caused by collusion, as opposed to STRIDE’s tree patterns that focus only on solo attackers. The results show that none of the subjects in the STRIDE session identified a possible collusion case between a client and servers, while 11 subjects in the ABC session identified this collusion case<sup>9</sup>. This confirms the importance of considering collusion when investigating threat cases, and shows the usefulness of ABC matrices in handling this task.

All these factors affected the overall correctness of the threat models built by the subjects. As shown in Figure 8, the ABC session scored an average of 64% as compared to 50% for STRIDE. This is expected based on the reported results for the modeling steps, where ABC scores were ahead of those from STRIDE.

**Framework Ease-of-Use** As mentioned previously, at the end of each session, we asked participants to report on how easy the framework in question was to apply. Ease-of-use was measured on a Likert scale in which 1 indicates the lowest value and 5 indicates the highest value. The average values are 3.9 for ABC and 3.8 for STRIDE<sup>10</sup>. This result becomes somewhat more significant when we point

<sup>8</sup> One participant in the ABC session has 0 false negative and 0 true positive values, so we excluded him/her when we computed the average.

<sup>9</sup> Most of them, however, did not provide a clear description of the attack scenario. Hence, these incomplete descriptions *were not* counted as correct threats when grading Step 3.

<sup>10</sup> Three participants did not complete the questionnaire in the STRIDE session.

out that the participants already had some exposure to STRIDE and its threat tree patterns. Even though it is a new framework that introduced several new concepts to the participants, ABC still achieved a comparable ease of use level. This suggests that participants were able to grasp its concepts through just a single hour of training, and therefore ABC shows potential as a usable method for threat modeling.

### 4.3 Threats to Validity

We acknowledge that a few limitations must be kept in mind when considering the study results. Empirical studies of threat modeling usually span a longer time frame, often on the orders of months, e.g., [66]. However, the fact that we were able to glean several important observations suggests that there may be lessons to be learned from short focused studies. In addition, we feel the design of our study might serve as a guide for determining promising areas for extended research before a large commitment of time and resources is made. Another constraining factor could be the age and experience level of our subjects. Different responses might have been obtained if we tested system security experts. However, we believe that the inexperience of our participants matches the cryptocurrency space well, which attracted users and researchers from various fields, even those from outside systems security. Therefore, our results give indications on how they might perform when investigating the security of cryptocurrencies.

## 5 Experiences

To demonstrate how ABC would function when applied to complex real-world systems, we developed use cases in which we built threat models for three cryptocurrencies, Bitcoin [63], Filecoin [36], and our system CacheCash. Furthermore, we report on a non-cryptocurrency use case that targets a cloud native security technology called SPIFFE, a project worked on by one of our authors. Each of these cases represents a different stage in a system design lifetime. Bitcoin and SPIFFE are well-established systems, Filecoin is under development and close to being launched, and CacheCash is still in its early development stages. The analyses for Bitcoin and Filecoin stopped before the risk management/mitigation phase. However, CacheCash and SPIFFE analyses involve the risk mitigation step as we describe later.

### 5.1 Bitcoin Analysis (Steps 1-3)

Bitcoin is by far the most valuable cryptocurrency with a capital market share of around \$245 billion as of late June 2019 [16]. As shown in Table 2, the Bitcoin threat model has significantly fewer collusion matrices and threat cases than other systems<sup>11</sup>. This is because it provides only a currency exchange service,

<sup>11</sup> The full threat model for Bitcoin is available online [32]

Table 2: Threat model comparison.

Aspect	Bitcoin	Filecoin	CacheCash	SPIFFE / SPIRE
ABC steps covered	Steps 1 - 3	Steps 1 - 3	Steps 1 - 4	Steps 1 - 4
Completion time (hr)	10	47	Not tracked	Not tracked
No. of collusion matrices	5	14	9	4
Total threat cases	105	882	525	1860
Distilled threat scenarios	10	35	22	65

which reduces the number of assets. Furthermore, it involves only two types of participants, miners and clients, which reduces the size of the collusion matrices. These factors, in addition to our familiarity with Bitcoin design details, contributed in reducing the completion time of Bitcoin’s threat model as shown in the table. Moreover, at the time we were working on this model, we had already completed the design of ABC. This suggests that deep understanding of the system model, and the availability of suitable tools impact not only the accuracy of the results, but also the time and effort expended in the threat modeling process.

We drew two main observations about the threat model we generated for Bitcoin. First, all the known threats to Bitcoin, such as double spending, Eclipse attacks [49], Goldfinger attack [55], and delaying blocks and transaction delivery [47], were mapped to the collusion matrices produced by ABC. Second, collusion between participants can play a major role in Bitcoin’s security. That is, several threats are neutralized by the assumption that at least 50% of the mining power is honest. Yet, mining pools have been formed to concentrate mining power. At the time of this writing, around 95% of the mining power is in the hands of just 10 mining pools [8]. If the managers of these pools decide to collude, they could break the honest majority barrier and take the system down. In fact, serious security attacks can be performed with less amount of mining power. Sompolinsky et al. [67] attest that a selfish mining attack, or blocks withholding, in which an attacker controls around 30% of the mining power, would be able to undermine the fairness of the mining reward distribution.

Furthermore, miner collusion may take different forms, such as rejecting specific updates on the network protocol. Cryptocurrencies are still in the development stage, and so their protocols continue to receive new features and updates. These updates can create forks in the network [40], and should a subset of the miners agree not to adopt the modified protocol, a new version of the currency may be spun off. This happened in Bitcoin, where two new cryptocurrencies split from its network including Bitcoin Cash [9] and Bitcoin Gold [10].

Usually, the >50% threat, or miners’ collusion in general, is argued about informally using incentive compatibility. This idea asserts that rational miners are more interested in keeping the system running to preserve the value of their

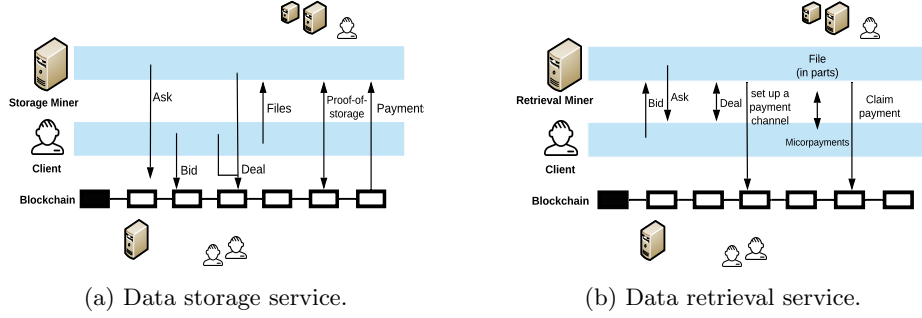


Fig. 9: Filecoin network model.

rewards. However, this claim is hard to verify and remains as an open question [42]. In addition, this assumption might be valid when all parties belong to the same system. Yet, miners could be working in several cryptocurrencies and it could be the case that destroying one to strengthen the other would be more profitable. Nonetheless, such observations highlight two key points. First, it indicates the importance of validating all the security assumptions a system makes in its design. And second, it points to the need for rational economic incentives to address some types of security threats that cannot be addressed by using only cryptographic approaches. The design of ABC accounts for such observations as mentioned earlier in this paper.

## 5.2 Filecoin Analysis (Steps 1-3)

Filecoin [36] is a cryptocurrency-based distributed file storage and retrieval system. Any party may join as a storage or retrieval miner to offer service to others. Filecoin operates distributed retrieval and storage markets where clients and miners can submit storage/retrieval bids and offers. Once these offers are matched, the service-payment exchange process, in which clients pay the miners in the Filecoin currency in exchange for receiving correct service, may start. The mining process in Filecoin is tied to the storage service miners put into the system. Recently, the Filecoin team raised around \$250 million through an ICO (initial coin offering) [1] in preparation for an official launch.

Filecoin is a more complicated system than Bitcoin as it provides two types of services on top of the currency exchange medium (these are captured by the network models shown in Figure 9). In addition, its protocol involves three participant roles: clients, retrieval miners, and storage miners, with the latter filling the traditional roles of miners in maintaining the blockchain. This complexity is reflected in the number of collusion matrices and threat cases produced as shown in Table 2. Moreover, all threat categories that target the service asset were replicated for each service type, which contributed to the large size of the threat model. These factors affected the completion time to build the model, which was 4.7x the time needed to build Bitcoin’s model. This cost in time

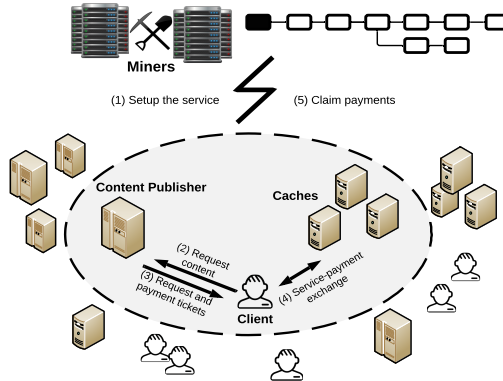


Fig. 10: CacheCash network model.

commitment is a natural result of working with newly developed and complex systems that provide a rich set of features.

In threat modeling Filecoin’s whitepaper, we found three unaddressed issues, mostly dealing with collusion cases that were not considered. Additionally, there are many places where the system is underspecified and so it is not possible to reason about whether or not it meaningfully addresses a threat.

*Ethics and disclosure.* We reached out to the Filecoin team, which mentioned efforts they have undertaken to resolve these problems. We withhold details about these issues until later as part of the responsible disclosure process.

### 5.3 CacheCash Analysis (Steps 1-4)

CacheCash is a cryptocurrency that provides a distributed content delivery service. It allows content publishers to construct dynamic networks of caches to serve their clients. This is done by allowing anyone to set up a new cache, and then collect cryptocurrency tokens from publishers for serving their clients (as captured by the network model shown in Figure 10). To handle the functionality of the currency exchange medium, CacheCash uses a modified version of the Bitcoin protocol.

We developed ABC during the early stages of our work on designing CacheCash. As the work progressed, we realized that most of the threat cases we encounter are related to the financial aspects of the system and the possible collusion between participants. Such aspects, as mentioned previously, are not explicitly addressed by traditional threat modeling frameworks. At that time we realized that none of these frameworks suited our needs, which lead to developing ABC.

As shown in Table 2, CacheCash’s threat model is smaller, in terms of the number of collusion matrices and threat cases, than the one developed for Filecoin. This is because CacheCash provides a single type of service on top of the

currency exchange medium, as compared to two in Filecoin. This is also reflected in the lower number of distilled threat cases than what Filecoin’s model produced.

Beyond threat modeling, we used ABC while designing threat mitigation techniques in the CacheCash system. During that time, we observed the importance of rational financial incentives in this process. This includes employing detect-and-punish mechanisms in which the penalty deposit of a party is revoked upon detecting that it is cheating, or designing algorithms that, when implemented in a malicious way, can cost the attacker more in resources than would working honestly. Furthermore, we realized the value of game theory and economic analysis in assessing the effectiveness of these economic threat mitigation approaches, and in quantifying the risk, or amount of damage, that financial attacks may cause. To date, we found ABC useful for CacheCash in both the pre-design threat modeling step, and the after-design security analysis of the system modules.

As CacheCash’s design is not public yet, its full threat model will be released at the public unveiling of the system in early 2020.

#### 5.4 SPIFFE/SPIRE Analysis (Steps 1-4)

The Secure Production Identity Framework for Everyone (SPIFFE) is a technology that targets the problem of obtaining identities in cloud-based distributed systems. SPIFFE, which is a project under the umbrella of the Linux Foundation, allows a service to acquire a secure identity that can be used to authenticate itself and authorize its access to other services and system resources. Thus, it solves the scalability problem of conventional security practices when working across heterogeneous environments and organizational boundaries. The SPIFFE Runtime Environment (SPIRE) is the production-ready implementation of SPIFFE APIs. Its work model, depicted in Figure 11, consists of a central server that mints identity documents for applications (or workloads), and node agents that perform workload attestation and distribution of IDs.

The SPIFFE/SPIRE team collaborated with one of the authors of this paper in using ABC to evaluate the framework’s security properties and to understand the potential risks in various practical deployment scenarios. The complete process can be found in [27, 28]. This use case is different from the previously discussed ones because it does not involve any form of explicit payment exchange between the participants. Thus, it attests to the applicability of ABC for not only cryptocurrencies, but also for large-scale distributed systems in general.

Through their analysis, the SPIFFE team members produced four threat categories: misrepresentation of identity, identity theft, compromise/remote code execution, and DoS. They also outlined the specific capabilities, or strength classes, an attacker may need to perform a given attack. Such an attacker could be the SPIRE server, a node agent, a workload (on the same node or on a different node), or even an external entity. Listing the attacker strength classes helped in both enumerating the possible threat scenarios, as well as in estimating the likelihood and severity of each distilled threat case.

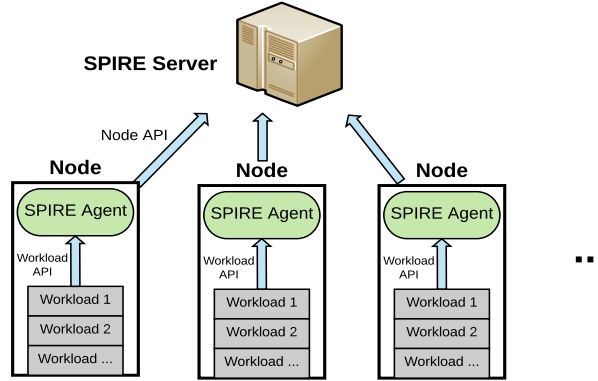


Fig. 11: SPIFFE/SPIRE network model.

As shown in Table 2, although the threat model of SPIFFE has the least number of collusion matrices, it produced the largest number of threat cases (around 17.7x, 2x, 3.5x of what Bitcoin, Filecoin, and CacheCash produced, respectively). The lower number of matrices is due to the fact that all other systems have the currency exchange medium, which adds another layer beneath the distributed service they provide. SPIFFE, on the other hand, provides only an identity production service that can be run as an additional layer on top of any cloud native application. However, SPIFFE has 4 participant roles (on the attacker side it would be 5 when counting the external entity). This made its collusion matrices larger, and hence, resulted in more threat cases than the other systems.

Similar to the CacheCash use case, ABC was used during the risk management and threat mitigation step. The team members computed a score for each of the distilled threat scenarios that estimates the likelihood that an attacker possesses the required capabilities, and the severity of the attack to the system. Studying the list of scored threat scenarios, the model outlined 54 unaddressed threat cases, most of which have very low scores, and produced several insights that guided the team on where to focus their efforts. Prior to the threat modeling process, the team envisioned several threat mitigation mechanisms that could complicate the system design. Once they determined that the impact of such attacks is very low, the higher score attacks were considered instead. These scenarios required even simpler mitigation techniques. For example, one type of DoS attacks can be mitigated using a rate limit control mechanism that the team integrated with the system [3].

Accordingly, this case shows how ABC can be very useful in assessing the security level of an already deployed system, defining which threat cases are already neutralized, and mitigating high profile, previously unaddressed, attacks.

## 6 Conclusions

In this paper, we introduce ABC, a cryptocurrency-focused threat modeling framework. Its design is motivated by the observation that traditional threat modeling frameworks do not fit cryptocurrencies, thus leaving them vulnerable to unanticipated attacks. ABC introduces collusion matrices, a technique that allows designers to investigate hundreds of threat cases in a reasonable amount of time. This is in addition to a flexible mechanism to derive system specific threat categories that focus on the assets to be protected and account for the financial motivations of the attackers. Both the user study and the use cases confirm that our framework is effective in unraveling hidden threat cases. The evaluated cases cover various types of cryptocurrency-based systems and a cloud native security technology under the Linux Foundation. This shows the potential of ABC to improve the security of a wide array of distributed systems.

## References

1. *\$257 Million: Filecoin Breaks All-Time Record for ICO Funding.* <https://www.coindesk.com/257-million-filecoin-breaks-time-record-ico-funding/>.
2. *\$400,000 stolen in Lumens BlackWallet theft.* <https://www.zdnet.com/article/400000-stolen-in-lumens-blackwallet-theft/>.
3. *Add ratelimiter for SPIRE node API.* <https://github.com/spiffe/spire/pull/577>.
4. *Benebit The Biggest ICO Exit Scam In History Nets Up to \$4 Million.* <https://www.coinbureau.com/ico/benebit-biggest-ico-exit-scam-history-nets-4-million/>.
5. *Binance cryptocurrency sell-off disaster blamed on mass phishing campaign.* <https://www.zdnet.com/article/binance-cryptocurrency-sell-off-disaster-blamed-on-mass-phishing-campaign/>.
6. *Bitcoin Cash exploit cripples network during scheduled hardfork upgrade.* <https://cryptoslate.com/bitcoin-cash-exploit-cripples-network-hardfork/>.
7. *Bitcoin Gold suffers double spend attacks, \$17.5 million lost.* <https://www.zdnet.com/article/bitcoin-gold-hit-with-double-spend-attacks-18-million-lost/>.
8. *Bitcoin mining pools.* <https://blockchain.info/pools>.
9. *BitcoinCash.* <https://www.bitcoincash.org/>.
10. *BitcoinGold.* <https://bitcoingold.org/>.
11. *The Bitfinex Bitcoin Hack: What We Know (And Don't Know).* <https://www.coindesk.com/bitfinex-bitcoin-hack-know-dont-know/>.
12. *Bitfloor Hacked, \$250,000 Missing.* <https://bitcoinmagazine.com/articles/bitfloor-hacked-250000-missing-1346821046/>.
13. *Bitstamp Claims \$5 Million Lost in Hot Wallet Hack.* <https://www.coindesk.com/bitstamp-claims-roughly-19000-btc-lost-hot-wallet-hack/>.
14. *Box plot diagrams.* <https://www.itl.nist.gov/div898/handbook/eda/section3/boxplot.htm>.
15. *CoffeeMiner hijacks public Wi-Fi users' browsing sessions to mine cryptocurrency.* <https://www.zdnet.com/article/how-to-hack-public-wi-fi-to-mine-for-cryptocurrency/>.



16. *CryptoCurrency Market Capitalizations*. <https://coinmarketcap.com/>.
17. *Enigmas Hack: \$500,000 of Ether Stolen, Accounts Compromised*. <https://cointelegraph.com/news/enigmas-hack-500000-of-ether-stolen-accounts-compromised>.
18. *Golem*. <https://golem.network/>.
19. *Hackers Hijack Another Ethereum ICO, Small Number of Users Affected*. <https://www.bleepingcomputer.com/news/security/hackers-hijack-another-ethereum-ico-small-number-of-users-affected/>.
20. *Hackers Stole \$32 Million in Ethereum*. <https://thehackernews.com/2017/07/ethereum-cryptocurrency-hacking.html>.
21. *Microsoft Threat Modeling Tool 2016 User Guide*. <https://www.microsoft.com/en-us/download/details.aspx?id=49168>.
22. *Monero*. <https://www.getmonero.org/>.
23. *Monex to revamp Coincheck cryptocurrency exchange in mere months*. <https://www.zdnet.com/article/monex-to-revamp-coincheck-cryptocurrency-exchange-within-two-months/>.
24. *One of the world's biggest bitcoin exchanges has been hacked*. <http://www.businessinsider.com/south-korean-bitcoin-exchange-bithumb-hacked-ethereum-2017-7>.
25. *Poloniex Loses 12.3% of its Bitcoins in Latest Bitcoin Exchange Hack*. <https://www.coindesk.com/poloniex-loses-12-3-bitcoins-latest-bitcoin-exchange-hack/>.
26. *Ripple*. <https://ripple.com/>.
27. *Scrutinizing SPIRE to Sensibly Strengthen SPIFFE Security: Part I, Methodology*. <https://blog.scytale.io/scrutinizing-spire-security-9c82ba542019>.
28. *Scrutinizing SPIRE to Sensibly Strengthen SPIFFE Security: Part II, Findings*. <https://blog.scytale.io/scrutinizing-spire-security-9c82ba542019>.
29. *South Korean cryptocurrency exchange hack sees \$40m in altcoin stolen*. <https://www.zdnet.com/article/south-korean-cryptocurrency-exchange-hack-sees-40m-in-altcoin-stolen/>.
30. *SPIFFE*. <https://spiffe.io/>.
31. *Steemit Hacked for \$85,000 as Users Complain of Weak Security*. <https://news.bitcoin.com/steemit-hacked-weak-security/>.
32. *Supplemental Material*. <https://ssl.engineering.nyu.edu/papers/abc-material.zip>.
33. *Understanding the DAO Attack*. <https://www.coindesk.com/understanding-dao-hack-journalists/>.
34. *Veritaseum Founder Claims \$8 Million in ICO Tokens Stolen*. <https://www.coindesk.com/veritaseum-founder-claims-8-million-ico-token-stolen/>.
35. *Zcash*. <https://z.cash/>.
36. *Filecoin: A Cryptocurrency Operated File Storage Network*, 2017. <https://filecoin.io/filecoin.pdf>.
37. ALBERTS, C. J., AND DOROFEE, A. *Managing information security risks: the OCTAVE approach*. Addison-Wesley Longman Publishing Co., Inc., 2002.
38. ANDEL, T. R., AND YASINSAC, A. Adaptive threat modeling for secure ad hoc routing protocols. *Electronic Notes in Theoretical Computer Science* 197, 2 (2008), 3–14.
39. ANDROULAKI, E., KARAME, G. O., ROESCHLIN, M., SCHERER, T., AND CAPKUN, S. Evaluating user privacy in bitcoin. In *Financial Cryptography and Data Security* (2013), pp. 34–51.

40. ANTONOPOULOS, A. M. *Mastering Bitcoin: Programming the open blockchain*. O'Reilly Media, Inc., 2017.
41. BACKES, M., KATE, A., MANOHARAN, P., MEISER, S., AND MOHAMMADI, E. Anoa: A framework for analyzing anonymous communication protocols. In *IEEE Computer Security Foundations Symposium* (2013), pp. 163–178.
42. BONNEAU, J., MILLER, A., CLARK, J., NARAYANAN, A., KROLL, J. A., AND FELTEN, E. W. Sok: Research perspectives and challenges for bitcoin and cryptocurrencies. In *IEEE S&P* (2015), pp. 104–121.
43. CHEN, Y., BOEHM, B., AND SHEPPARD, L. Value driven security threat modeling based on attack path analysis. In *2007 40th Annual Hawaii International Conference on System Sciences (HICSS'07)* (2007), IEEE, pp. 280a–280a.
44. CLAYCOMB, W. R., AND SHIN, D. Threat modeling for virtual directory services. In *IEEE ICCST* (2009), pp. 149–154.
45. DENG, M., WUYTS, K., SCANDARIATO, R., PRENEEL, B., AND JOOSEN, W. A privacy threat analysis framework: supporting the elicitation and fulfillment of privacy requirements. *Requirements Engineering* 16, 1 (2011), 3–32.
46. GARAY, J., KIAYIAS, A., AND LEONARDOS, N. The bitcoin backbone protocol: Analysis and applications. In *EUROCRYPT* (2015), pp. 281–310.
47. GERVAIS, A., RITZDORF, H., KARAME, G. O., AND CAPKUN, S. Tampering with the delivery of blocks and transactions in bitcoin. In *ACM CCS* (2015), pp. 692–705.
48. HASAN, R., MYAGMAR, S., LEE, A. J., AND YURCIK, W. Toward a threat model for storage systems. In *ACM workshop on Storage security and survivability* (2005), pp. 94–102.
49. HEILMAN, E., KENDLER, A., ZOHAR, A., AND GOLDBERG, S. Eclipse attacks on bitcoin's peer-to-peer network. In *USENIX Security Symposium* (2015), pp. 129–144.
50. HOLICK, M., NITA-ROTARU, C., PAPADIMITRATOS, P., PERRIG, A., AND SCHMID, S. Toward a taxonomy and attacker model for secure routing protocols. *ACM SIGCOMM Computer Communication Review* 47, 1 (2017), 43–48.
51. HOWARD, M., AND LIPNER, S. *The security development lifecycle*, vol. 8. Microsoft Press Redmond, 2006.
52. JAVAID, A. Y., SUN, W., DEVABHAKTUNI, V. K., AND ALAM, M. Cyber security threat analysis and modeling of an unmanned aerial vehicle system. In *2012 IEEE Conference on Technologies for Homeland Security (HST)* (2012), IEEE, pp. 585–590.
53. KAPPOS, G., YOUSAF, H., MALLER, M., AND MEIKLEJOHN, S. An empirical analysis of anonymity in zcash. In *27th {USENIX} Security Symposium ({USENIX} Security 18)* (2018), pp. 463–477.
54. KOSBA, A., MILLER, A., SHI, E., WEN, Z., AND PAPAMANTHOU, C. Hawk: The blockchain model of cryptography and privacy-preserving smart contracts. In *2016 IEEE symposium on security and privacy (SP)* (2016), IEEE, pp. 839–858.
55. KROLL, J. A., DAVEY, I. C., AND FELTEN, E. W. The economics of bitcoin mining, or bitcoin in the presence of adversaries. In *WEIS* (2013).
56. LUNA, J., SURI, N., AND Krontiris, I. Privacy-by-design based on quantitative threat modeling. In *IEEE CRiSIS* (2012), pp. 1–8.
57. LUU, L., CHU, D.-H., OLICKEL, H., SAXENA, P., AND HOBOR, A. Making smart contracts smarter. In *ACM CCS* (2016), pp. 254–269.
58. LUU, L., TEUTSCH, J., KULKARNI, R., AND SAXENA, P. Demystifying incentives in the consensus computer. In *ACM CCS* (2015), pp. 706–719.

59. MORENO-SANCHEZ, P., MODI, N., SONGHELA, R., KATE, A., AND FAHMY, S. Mind your credit: Assessing the health of the ripple credit network. *arXiv preprint arXiv:1706.02358* (2017).
60. MORENO-SANCHEZ, P., ZAFAR, M. B., AND KATE, A. Listening to whispers of ripple: Linking wallets and deanonymizing transactions in the ripple network. *Privacy Enhancing Technologies*, 4 (2016), 436–453.
61. MÖSER, M., SOSKA, K., HEILMAN, E., LEE, K., HEFFAN, H., SRIVASTAVA, S., HOGAN, K., HENNESSEY, J., MILLER, A., NARAYANAN, A., ET AL. An empirical analysis of traceability in the monero blockchain. *Proceedings on Privacy Enhancing Technologies 2018*, 3 (2018), 143–163.
62. MYAGMAR, S., LEE, A. J., AND YURCIK, W. Threat modeling as a basis for security requirements. In *Symposium on requirements engineering for information security* (2005), pp. 1–8.
63. NAKAMOTO, S. Bitcoin: A peer-to-peer electronic cash system.
64. PARNO, B., HOWELL, J., GENTRY, C., AND RAYKOVA, M. Pinocchio: Nearly practical verifiable computation. In *IEEE S&P* (2013), pp. 238–252.
65. PASS, R., SEEMAN, L., AND SHELAT, A. Analysis of the blockchain protocol in asynchronous networks. In *EUROCRYPT* (2017), pp. 643–673.
66. SCANDARIATO, R., WUYTS, K., AND JOOSEN, W. A descriptive study of microsofts threat modeling technique. *Requirements Engineering* 20, 2 (2015), 163–180.
67. SOMPOLINSKY, Y., AND ZOHAR, A. Secure high-rate transaction processing in bitcoin. In *Financial Cryptography and Data Security* (2015), pp. 507–527.
68. TADELIS, S. *Game theory: an introduction*. Princeton University Press, 2013.
69. TORR, P. Demystifying the threat modeling process. *IEEE Security & Privacy* 3, 5 (2005), 66–70.
70. VAN LAMSWEERDE, A. Elaborating security requirements by construction of intentional anti-models. In *Proceedings of the 26th International Conference on Software Engineering* (2004), IEEE Computer Society, pp. 148–157.
71. VANDERVORT, D., GAUCAS, D., AND ST JACQUES, R. Issues in designing a bitcoin-like community currency. In *Financial Cryptography and Data Security* (2015), pp. 78–91.
72. WOOD, G. Ethereum: A secure decentralised generalised transaction ledger.

## A Deriving ABC Threat Categories

In this section, we show how the threat categories listed in Table 1 were derived. We apply the procedure outlined in Step 2 of the ABC framework (Section 3.2) to the assets of CompuCoin, which include the service (e.g., computation outsourcing in case of CompuCoin), service rewards or payments, blockchain, transactions, currency, and communication network. For each asset, we outline its security properties in order to identify any factors that might violate these properties. These factors are labeled as threat categories.

Starting with the **service asset**, our analysis outlines the following:

1. *Security properties*: A secure service can be defined as the action of serving clients correctly at anytime, while providing confidentiality and binding the servers to the service they provide. Hence, the security properties of a service asset may include integrity, availability, confidentiality, and non-repudiation.

2. *Threat categories:* By negating the above properties, we find that the service asset has the following threat categories:
  - Service tampering/corruption: An attacker provides clients with invalid service or corrupts the correct service delivered by others.
  - Information disclosure: An attacker reveals the contents of service-related messages, such as the service content/outcome, the service requests sent by clients, replies sent by servers, etc.
  - Repudiation: A server denies providing a specific service or a client denies receiving it.
  - DoS: An attacker makes the service unavailable to legitimate users.

Next, we analyze the **service payment** (or rewards) asset as follows:

1. *Security properties:* One may consider the service payment asset secure as long as: a) servers are rewarded properly for the service they provide, and b) servers earn the payments they collect.
2. *Threat categories:* Negating the above security requirements produce the following threat categories:
  - Service slacking: A server collects payments without performing all the promised work.
  - Service theft: A client obtains service for a lower payment than the agreed upon amount.

For the **blockchain asset**, our analysis produces the following:

1. *Security properties:* The blockchain security properties are tied to the security of the underlying consensus protocol. These properties have been thoroughly studied in the literature [42, 46, 65]. We adopt the ones introduced in [65] with slight modifications based on the work presented in [42], which include:
  - Consistency: At any point in time, honest miners hold copies of the blockchain that have a common prefix and may differ only in the last  $y$  blocks, where  $y$  is a block confirmation parameter. A block then is confirmed once it is buried under  $y$  blocks on the blockchain.
  - Future-self consistency: At any two points in time,  $t_1$  and  $t_2$ , the blockchain maintained by an honest party may differ only in the last  $y$  blocks. Consistency and future-self consistency properties achieve blockchain persistence or immutability.
  - Fairness: Miners collect mining rewards in proportion to the resources they expend in the mining process.
  - Correctness: All the blocks within the longest branch in the blockchain are valid. (Note that correctness and fairness represent the chain quality property outlined in [46, 65].)
  - Growth: As long as the system is functional, new valid blocks will be added to the blockchain.
2. *Threat Categories:* By negating the above properties, we can distill the following threat categories for the blockchain asset:

- Inconsistency: Honest miners do not agree on the prefix of the blockchain copies they hold beyond the unconfirmed blocks. This also covers the case of an honest miner who does not agree with itself on the blockchain prefix it holds over time, e.g., alternating between two branches that compete in being the longest.
- Invalid block adoption: The longest chain contains corrupted blocks that either have an invalid format or contain invalid transactions.
- Biased mining: A miner pretends to expend the needed resources to be selected to extend the blockchain and collect the mining rewards.
- Chain freezing: The blockchain does not grow at a regular rate, but instead freezes for several contiguous rounds. This threat category is a form of DoS attack, and hence, we cover it under DoS against the communication network asset.

Next we analyze the **transaction asset** as follows:

1. *Security properties*: Secure transactions can be characterized as correct, tamper-proof, and source-binding, i.e., cannot be denied by the originator. In addition, these transactions need to be accessible to the system users at any time so they can send/receive/view transactions as needed. Moreover, these transactions must not reveal any information about the source, destination, and amount of transferred funds. Accordingly, we outline the following security properties for the transaction asset: non-repudiation, integrity, validity, availability, and anonymity. Note that the validity property is already covered by the correctness aspect of the blockchain, where a valid blockchain contains only valid transactions. Furthermore, the availability property is covered under the communication network asset.
2. *Threat categories*: Based on the previous discussion, and again by negating the aforementioned security properties, the threat categories for the transaction asset would be:
  - Repudiation: An attacker denies issuing transactions.
  - Tampering: An attacker manipulates the fields of a transaction.
  - Deanonymization: An attacker violates users' privacy by exploiting the public nature of the blockchain to link transactions and payments, and use this knowledge to track the activity of these users in the system and, possibly, reveal their real identities.

Next we analyze the **currency asset** as follows:

1. *Security properties*: The security properties of the currency asset are intertwined with the properties of the transaction asset. This is because the currency takes the form of digital tokens, which are the transactions exchanged in the system. Thus, they inherit all the transaction security properties. What remains is to deal with the currency ownership, meaning that only the owner can spend these tokens.
2. *Threat categories*: Beside the categories outlined above for the transaction asset, we have the following threat category for the currency asset:

- Currency theft: An attacker steals currency from others in the system. This includes all currency theft attacks that are not covered by other assets. For example, biased mining, where a miner steals others rewards indirectly, is currency theft, but it is already covered by the blockchain. The same holds true for the service payment related threats.

Finally, we analyze the **communication network asset** as follows:

1. *Security properties:* The communication network is the backbone of any cryptocurrency system, and one that is unreliable can lead to numerous problems. First, it can create delays in propagating newly mined blocks that could produce an inconsistent blockchain. Second, it can cause delays in relaying transactions, which could reduce the transaction throughput of the system and affect its availability aspect. Third, it can slow down setting up new miners who need a longer time to discover other peers and download copies of the blockchain. Fourth, it opens the possibility of being controlled by external parties that could intercept the communication links and isolate nodes in the network. Consequently, a secure cryptocurrency system needs a reliable and robust communication network. We merge all these aspects into one security property, namely, availability.
2. *Threat categories:* The communication network asset has one threat category, which is DoS.

Table 1 summarizes all the threat categories derived in this appendix. As mentioned previously, this table is by no means comprehensive. Additional threats can be added based on the asset types of the system, or more refined definitions of the asset security properties. This detailed treatment was provided as a thorough example to clarify the application of Step 2 in the ABC framework. Nonetheless, we found this table sufficiently detailed when building threat models for the systems reported as use cases in Section 5, including Bitcoin, Filecoin, and CacheCash, as well as for the user study tutorial as reported in Section 4. For this reason, Table 1 can be viewed as a base threat list that can be extended, or even reduced, based on the system under design.