Security Survey of Bitcoin

Ruven Chu and Andrew He
Stanford University

Abstract

We present our methodology for analyzing the security properties of the Bitcoin cryptocurrency. We use the formal verifier tool Murφ to investigate the double-spending problem, finding two known attacks using state enumeration. We briefly discuss the lack of true anonymity in Bitcoin transactions as well as Bitcoin's robustness against denial-of-service attacks.

1 Introduction

Bitcoin is a decentralized peer-to-peer digital currency intended to allow secure transactions without any central authority [Nak]. The Bitcoin system adopts many security mechanisms in response to typical attacks on electronic currency, which include stealing, counterfeiting, and double-spending. To prevent theft or illegal duplication of Bitcoins, the system ensures the integrity of each transaction via digital signatures (ECDSA). To prevent double-spending (in which a single coin is used simultaneously for multiple transactions), Bitcoin implements something similar to a distributed timestamp server: every node's transactions are broadcast to the entire network, such that all nodes in the network maintain a public chain of previous transactions. Transactions involving a specific coin are not added to this chain (i.e. *approved*) until the "unused-ness" of the coin is verified. In addition to being robust against illegal transactions, Bitcoin aims to support anonymous transactions. We use intuitive analysis and finite-state model checking in order to confirm whether the Bitcoin specification is sufficient for maintaining these security properties.

2 Security properties

We focus primarily on two of the security properties detailed in the Bitcoin design document – protections for double-spending and privacy – as well as another typical property of peer-to-peer networks: robustness against denial-of-service attacks. Because the cryptography schemes used in Bitcoin are based on federal standards (SHA-256 and ECDSA), our model is designed with idealized cryptography [BM] in mind; therefore, security properties involving coin integrity (e.g. robustness against theft or counterfeiting) are already assumed to

hold. In addition to verifying these security properties, we also seek to evaluate them from a practical standpoint (§4, 5, 6).

Robustness against double spending. Once a user spends a coin in a certain transaction, that user cannot spend the same coin in a different transaction. We attempt to verify this security property using finite-state model checking.

Privacy (anonymity). The system should not leak information about the participants of transactions, other than the participants' public keys.

Robustness against denial-of-service. A user should be able to send and receive transactions with bounded waiting time. We investigate this and the previous property by vetting the system by hand.

3 Modeling Bitcoin

We construct a basic model of the Bitcoin system using the Murφ protocol verification tool [Dil]. Murφ performs state enumeration and automatically checks whether each reachable state satisfies user-specified invariants. In our case, we only use Murφ to verify the double-spending property, so our main invariant is that property. We describe Bitcoin in the Murφ language as a system involving honest agents (Participators) and attackers (Intruders). Messages between these agents are simply transaction broadcasts: Participators are modeled to have perfect broadcast capabilities, whereas the Intruder can be an imperfect broadcaster. To derive this rule, we note that while honest nodes forward transactions as they hear them, attackers may intercept transactions on their way to their recipients or refuse to forward any/all transactions. In other words, attackers follow the Dolev-Yao model [Cer] and are subject only to cryptography constraints.

In formulating the message model between Participators and Intruders, we simplify the system by leaving out finer details that are not pertinent to the security properties:

- Protocol abstraction: Messages between nodes only contain transaction data. Handshaking/connection messages and query messages are not modeled.
- Coin generation abstraction: Each node begins with a fixed number of coins. There is no coin generation.
- Block abstraction: In the Bitcoin system, one or more transactions are typically gathered into blocks for batch approval. In our simplified model, each block only contains a single transaction, so the block chain is just a tree of transactions.

We also include in our model the notion of "work", which represents the number of transactions each agent is allowed to approve during a single run of the model-checker. In the real Bitcoin system, this is analogous to the amount of CPU

power a Bitcoin node possesses, which roughly determines how quickly the node can produce a proof-of-work for a block. This abstraction is important, as it allows us to investigate the sensitivity of our security properties to the balance of power in the network.

Murφ was run with varying parameters: we adjusted the numbers of Participators and Intruders, as well as the amount of work available to each. In the process, we discovered several violations of the security invariant, although all of these were related to previously-known attacks. We describe our findings in the next section.

4 The double-spending problem

We report two types of double-spending attacks, based on information from our Murφ traces. The first involves network segmentation, in which one or more attackers successfully split the network graph into two non-communicating subgraphs. The second attack involves malicious nodes controlling a majority of the network's computing power. Both attacks have been discussed in previous literature.

4.1 Segmentation

If the nodes in the network are divided into two non-communicating subgraphs, double-spending can easily occur. We provide a simple example. Suppose that an attacker (Mallory) places a set of attack nodes between a victim (Bob) and the rest of the network. If Bob has no other connection to the network other than Mallory's nodes, Bob might be incapable of garnering approvals for his own transactions, or he may not be able to hear transactions originating from the rest of the network. Mallory can then craft two transactions using the same coin: one to Alice (some node in the network) and one to Bob. She can broadcast $Transaction_{Alice}$ to the network, while only sending $Transaction_{Bob}$ to Bob. $Transaction_{Alice}$ is approved by the rest of the network and added to the block chain. Meanwhile, either Bob or one of Mallory's attack nodes approves $Transaction_{Bob}$, causing Bob to add it to his own chain (i.e. he believes the coin was given to him). Thus, Mallory has successfully spent the same coin in multiple transactions. Once Bob regains connectivity with the network, he will be informed of this discrepancy; however, Mallory may have already received goods or services from Bob by that time.

In practical terms, we find that a Bitcoin node is vulnerable to segmentation if a network attacker can successfully interpose on that node's peer discovery messages. Currently, nodes who wish to find peers first connect (in an insecure fashion) to an IRC channel, through which they can learn peers' IP addresses. An attacker who spoofs the IRC communications can easily convince a victim to connect to attack nodes, rather than honest nodes. In fact, other Bitcoin peer discovery techniques are similarly vulnerable, since communications are typically unencrypted.

4.2 Majority power

Once an attacker controls a majority of the network's CPU power, it can trivially modify the ordering of transactions occurring thereafter. Honest nodes always assume that the longest block chain is valid, but the attacker is clearly capable of extending the block chain faster than the honest network. Hence, the attacker can theoretically perform the following actions:

- Refuse to add transactions to the history
- Approve one of its own transactions, and then quickly erase that transaction from history (*double-spending*)

It is worth noting that this type of double-spending attack is addressed in the original Bitcoin design document, and dismissed on the basis that such an attacker would be incentivized to generate coins like an honest participant. This statement is largely true, but there exist scenarios in which the potential for manipulating large-sum payments could motivate malicious actions.

5 Anonymity

Bitcoin provides *pseudonymity* – its users are only referenced via random account addresses (their public keys). In order for Bitcoin to be truly anonymous, however, these public keys need to be unlinkable to real identities. We observe that for many practical purposes, Bitcoin makes it difficult, if not impossible, for a user to remain truly anonymous.

The danger of linked addresses lies in the fact that recovering all the transactions corresponding to a Bitcoin address is easy, since the transaction history is completely public. If a *stalker* wants to be able to link Bitcoin addresses to actual identities, one approach is to become a merchant of tangible goods. Upon receiving payment from a Bitcoin address, the merchant can request a physical shipping address from the buyer, thus associating the Bitcoin and physical addresses. The buyer's degree of anonymity then depends on whether his physical address leaks any information about him. Note that the merchant could also retrieve other information about the buyer (e.g. email / IP address) depending on the initial communication channel.

One suggested approach for improving anonymity is to never use the same public key twice. This is slightly complicated as coins can only be sent from the address with which they were received. If a user creates a new address, but transfers coins to it from an un-anonymized address, a stalker can notice and simply look up any transactions made from the new address. The solution is to employ an external mixer – a service provider that gathers coins from multiple source addresses and randomly redistributes them to destination addresses, essentially providing a stalker with spurious address associations. However, this solution has several vulnerabilities: the mixer could be incorrectly implemented or compromised, or network attackers could still link old / new addresses using IPs.

It seems that idea of a distributed system with public transaction records (such as Bitcoin) may inherently conflict with the concept of truly anonymous currency. One possible proposal for reconciling the two involves using Bitcoins as a base currency for an anonymous form of money, perhaps based on David Chaum's idea of blind signatures [Don08].

6 Denial of service

Bitcoin avoids having a single point of failure by definition, but based on our analysis of the segmentation double-spending attack (§4.1), it follows trivially that individual users are vulnerable to denial of service. This is not an inherent fault of Bitcoin, as an active attacker can deny any network service it chooses. Nevertheless, the risk of any particularly node becoming "surrounded" by attackers might be mitigated with the adoption of a secure peer discovery mechanism, as stated previously.

7 Conclusions

This paper reports the results of a security study of the Bitcoin system using a combination of finite-state model-checking and traditional analysis by hand. Using Murφ error traces, we confirmed a few known double-spending vulnerabilities in Bitcoin, and illustrated practical attacks based on these vulnerabilities. We also discussed potential improvements to increase privacy and resilience to denial-of-service attacks on individual users.

References

[BM] Jason Bau and John C. Mitchell, *Security modeling and analysis.*

[Cer] Iliano Cervesato, *The dolev-yao intruder is the most powerful attacker.*

[Dil] David L. Dill, *Murphi description language and verifier.*

[Don08] James A. Donald, *Re: Bitcoin p2p e-cash paper*, Email, November 2008.

[Nak] Satoshi Nakamoto, *Bitcoin: A peer-to-peer electronic cash system.*