CISC 322

Assignment 3 Report

BitCoin Core: Secure Digital Wallet

April 12th, 2023

Group 44 :

Victor Coombes (20vmwc@queensu.ca)

Osamu Adun (19osa2@queensu.ca)

Anton Gudonis (19adg8@queensu.ca)

Selena Kesidis (21smk13@queensu.ca)

Navin Pandey (17np1@queensu.ca)

Kyle Verma (18kv10@queensu.ca)

**Table of Contents**

**Abstract Pg 03**

**Introduction and Overview Pg 03**

**Enhancement Proposal Pg 04**

**Implementation 1 Pg 05**

**Implementation 2 Pg 07**

**SAAM Pg 10**

**Use Cases Pg 12**

**Plan for Testing Pg 13**

**Potential Risk Pg 13**

**Naming Conventions Pg 13**

**Lessons Learned Pg 14**

**Conclusions Pg 14**

**References Pg 15**

**Abstract**

This report proposes the implementation of a hybrid Proof-of-Stake (PoS) consensus algorithm into the Bitcoin Core architecture. The proposed PoS implementation aims to augment the security, efficiency, and sustainability of the Bitcoin network. The PoS algorithm would involve participants staking their bitcoins as collateral, increasing their chances of being selected as validators to verify transactions. We present two possible implementations of PoS and conduct a SAAM analysis to determine the most effective implementation. The proposed hybrid PoS implementation has the potential to enhance the consensus mechanism of Bitcoin Core, offering a promising approach for the evolution of the network.

**Introduction and Overview**

Throughout this semester, we conducted a thorough examination of Bitcoin Core’s software architecture. We found that the system operates in a peer-to-peer architecture style that integrates elements of both publish/subscribe and client-server styles in the system to provide users with a first-party, open-sourced wallet solution that allows users to mine bitcoin if desired. This discovery came from identifying the conceptual architecture to gain insight on what the product aims to accomplish and examining the concrete architecture to gain a comprehensive view of the system at work.

From our findings in the conceptual and concrete architecture reports, it has been deduced that there are 13 essential components within Bitcoin Core’s main system: App/GUI, RPC, wallet, Storage Engine, Headers, Blocks, Coins, Validation Engine, Mempool, Miner, Network, Utils and Performance Evaluation. These each play a fundamental role in Bitcoin Core’s system and it is crucial that through its enhancement, there are no radical shifts in these component’s code or interactions.

Upon examining Bitcoin Cores conceptual and concrete architecture, we have gained a comprehensive understanding of its design principles and functionality. Based on our analysis we propose a hybrid Proof-of-Stake (PoS)/Proof-of-Work (PoW) consensus algorithm to address the vulnerabilities of each algorithm while maximizing their benefits. Specifically, the proposed algorithm would mitigate the vulnerability of PoW to 51% attacks while maintaining the overall security and decentralization. This is achieved through requiring users to stake their Bitcoins as collateral to secure the opportunity to validate transactions and create new blocks, increasing validation rights in proportion to the amount of coins staked. Along with supplementing the existing PoW algorithm, users who engage in PoS validation would be rewarded with roughly 30% of the block reward as compensation.

Proof of Stake and Proof of Work are the two consensus algorithms used by blockchain networks in order to validate transactions and add new blocks to the chain. However, they differ in several drastic ways. PoW requires bitcoin miners to solve complex mathematical problems; the first one to solve it gets to add the next block to the chain and receive a reward. This requires a large amount of computational power and energy. On the other hand, PoS allows validators who hold a certain amount of cryptocurrency to participate in the consensus process, validating transactions and adding new blocks to the chain. By combining both these algorithms to enhance Bitcoin Core, it is hoped that users can experience the best qualities of both.

In order to implement a PoS algorithm into Bitcoin, we must fundamentally change its system architecture at the top level. Due to the fact that the top level subsystem, Validation Engine already contains a large quantity of the algorithmic code necessary, this subsystem can be built upon in order to fully implement PoS. However, despite resulting in minimal structural change, this can result in a complex and hard-to-maintain file structure. Alternatively, to implement PoS, two entirely new modules can be inserted into the system for maintaining the PoS algorithm and Forgers. This approach would strengthen the systems maintainability and modifiability by highlighting a separation between PoS and PoW logic. Both options for implementation have their own advantages and challenges which will be further examined in this report.

Overall, this report highlights the importance of maintaining a system architecture that is both easy to maintain and modify. Having gained a thorough understanding of Bitcoin Core’s system architecture, we were able to easily determine how new components or functions could be implemented within the software and further evaluate their impact on the system’s high level architecture. It would be much more complex to do so if Bitcoin Core’s software architecture was not designed optimally, with various functions divided into a proportionate amount of components and interactions. As a result, it is important that this mutability is inherited throughout Bitcoin Core’s evolution. To accomplish this, it has been determined that the PoS/PoW algorithm would be implemented by inserting the two new subsystems instead of modifying the Validation Engine. This will allow for Bitcoin Core to be improved upon without sacrificing any security or functionality within the system.

**Enhancement Proposal**

The implementation of a hybrid Proof-of-Stake/Proof-of-Work consensus algorithm into Bitcoin Core would introduce a new form of achieving validity on transactions and the creation of new blocks. In this proposed system participants would stake their bitcoins as collateral, the more coins are staked the more likely the participant is to earn the right to validate the block/transaction in question. This would supplement the existing PoW algorithm.

The purpose of such an enhancement is to minimize the downfalls of either algorithm while maximizing its upsides. The PoW consensus algorithm is inherently vulnerable to the 51% attack. PoS suffers from situations such as “nothing at stake” and low operational costs for miners may cause hoarding situations. Combining both consensus algorithms such that PoW miners continue their operations as normal, but when a block is found a secondary validation is done through some amount of PoS miners who are then rewarded around 30% of the block reward as compensation. This strategy limits the power of PoW miners to attack the blockchain maliciously while preventing PoS miners from hoarding coins and not giving PoS miners the option to perform “nothing at stake”. In addition, PoS can promote further decentralization by allowing more participants into the consensus process as PoS does not require intensive computational power like PoW, promoting the original vision of bitcoin.

**Current states:**

Since we added PoS/PoW as a component to Bitcoin Core, the previous conceptual architecture is now outdated and needs to be changed to include our component. These potential changes will impact the Validation Engine and all subsystems that it interacts with. Both of the proposed implementations impact the Validation Engine, which in turn impacts everything connected. The implementation alters the function of the Validation Engine to use PoW and PoS checks instead of what the system did prior. Two figures found below visualize both proposed implementations of our enhancement.

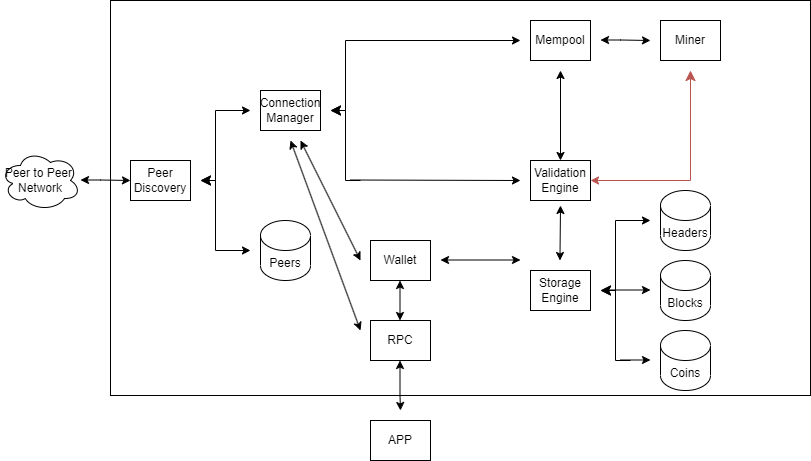
**Implementation 1**

The top level subsystem Validation Engine already houses most of the consensus algorithm code so logically implementing PoS by building upon this subsystem makes the most sense. By building upon this subsystem unexpected interactions can be avoided and a clean conceptual architecture can be maintained. Although, files within other subsystems must be updated or created to fully implement the functionality of PoS.

By building on top of the already existing subsystem and files, no new modules need to be created to support the PoS algorithm. The main subsystems/files affected by this implementation are the Validation Engine and App, although other subsystem files are also modified slightly to support the algorithm. Validation Engine is built upon such that primary PoW consensus files will now contain code/information regarding both algorithms. Files regarding miners will also be expanded on in the same way.

**High-level interactions:**

Due to building on top of the existing architecture and file system and the vast similarities of high level interactions between both consensus algorithms not many new high level interactions are present. However, the interaction between Validation Engine and Miner subsystems becomes bi-directional. This is due to how the PoS algorithm requests information from the miners directly to validate blocks. Compared to PoW where miners post their work to the Mempool from which the PoW algorithm can confirm the block/transaction on its own, at which point it notifies the miner whether the block/transaction was considered valid.



(Conceptual Architecture with first implementation changes added)

**Low-level interactions:**

A new low level interaction created by this implementation usage is a differentiator function call from Mempool to Validation Engine that determines which consensus algorithm is being called as both consensus algorithms are found within the same file. This is handled by the new file consensu\_manager.cpp which coordinates the PoS and PoW components. Message passing between Mempool and Validation Engine will also be changed as a result of this implementation. Previously upon validation using PoW, the Validation Engine would immediately post the transaction/block. Now with the inclusion of PoS message passing must be done to ensure both consensus algorithms agree. This is handled by the new file consensus\_manager.cpp.

With the implementation of an additional consensus algorithm rewards allotted to participating miners must be calculated differently. Therefore the file reward\_calculator.cpp is proposed to ensure block rewards are fairly distributed between PoW participants and PoS participants compared to when 100% of the rewards would go to the PoW individuals.

**Impact on current directories/files:**

New files:

consensus/consensus\_manager.cpp

reward\_calculator.cpp

Modified files:

Validation.cpp

Validation.h

consensus/params.cpp

consensus/consensus.h

consensus/tx\_verify.cpp

qt/bitcoin.cpp

qt/bitcoin.h

qt/overviewpage.cpp

**Considerations and challenges:**

Implementing PoS into Bitcoin Core will require the network to accept and validate the update. Coordinating a consensus upgrade can be challenging

PoS introduces a different set of security considerations compared to PoW. Although the implementation is meant to increase security, ensuring the robustness of the PoS mechanism is crucial.

Ensuring backwards compatibility through this implementation may prove difficult as crucial consensus files are updated directly.

**Thoughts:**

This proposed implementation results in minimal structural change to the overall architecture of Bitcoin Core but as a trade off results in an increasingly complex file structure where previously specialized files now perform two different algorithms.

**Implementation 2**

In an effort to separate the implementation of PoS into Bitcoin Core, it is proposed that two new modules be created. A system for maintaining the PoS algorithm and Forgers to represent participants in the PoS system. This option prioritizes maintainability and evolvability as the PoS algorithm will be distinct within the codebase, making it easy to identify and work with.

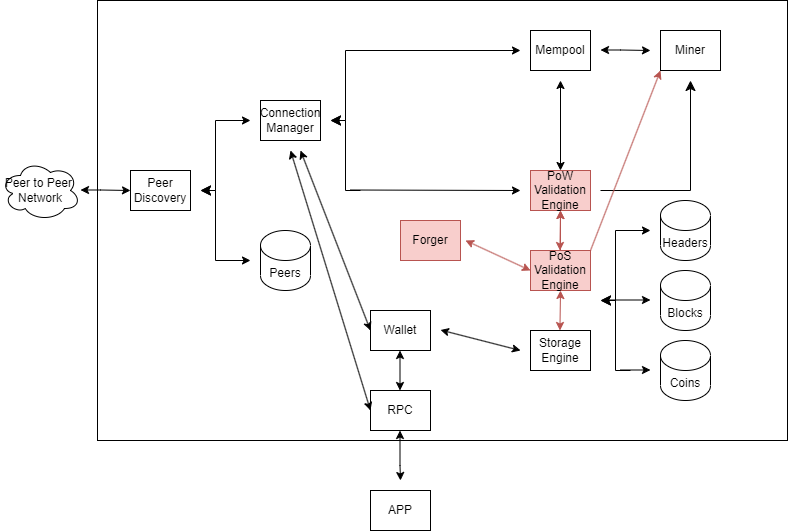
**Main changes:**

PoS algorithm: This is a new module to be added to the software architecture. It will be formatted similarly to the current validation engine subsystem and take on nearly the same responsibilities. Although due to the inherent difference in the algorithm execution the high level interactions between the two may vary. In addition the order of execution affects the required high-level interactions.

Forgers: This is a new module to be added to the software architecture. It will be formatted similarly to the miner subsystem and take on nearly the same responsibilities with a few exceptions. Forgers will primarily differ from Miners through their bi-directional interaction with the PoS algorithm to achieve validation/consensus.

**High-level interactions:**

With the introduction of two new high-level modules multiple new interactions must be discussed. The PoS Validation Engine will place itself in between the PoW Validation Engine and Storage Engine. This is because PoW Validation Engine no longer gets the final say on what gets posted to the blockchain. PoS Validation Engine has a unidirectional interaction with the Miner component. This is to allow the miner to be notified if the PoS validation determines the block/transaction is invalid. Furthermore, PoW Validation Engine has a unidirectional interaction with PoS Validation Engine. This is to represent the passing of a transaction/block validated by the PoW algorithm and moving onto the next step of the validation process. In addition, the PoS validation engine requires interaction with the storage engine not only to post blocks/transactions but to also retrieve information regarding stakes required for the decision making of the algorithm.



(Conceptual Architecture with second implementation changes added)

**Low-level interaction**

A key low-level interaction is between PoW Validation Engine and PoS Validation Engine through data exchange. PoW Validation Engine is required to send the block/transaction currently under validation to the next step of the process and a file is required to handle that. A similar low-level interaction is present between the PoS Validation Engine and the Storage Engine for the purposes of posting transactions/blocks to the network.

To maintain the security of the application a serialization function is required to obscure participants' data. Both ingoing and outgoing information regarding forgers is required to be serialized to protect the privacy of participants within Bitcoin Core.

**Impact on current directories/files:**

New files:

pow\_blockchain.cpp

pos\_validator.cpp

pow\_miner.cpp

pos\_staker.cpp

pos\_reward\_calculator.cpp

pow\_consensus\_algorithm.cpp

pos\_consensus\_algorithm.cpp

pos\_voting\_contract.cpp

data\_exchange.cpp

serialize.cpp

Modified Files(names changed)

pow\_Validation.cpp

pow\_Validation.h

pow\_consensus/params.cpp

pow\_consensus/consensus.h

pow\_consensus/tx\_verify.cpp

**Considerations and challenges:**

Implementing PoS in this way requires a restructuring of the order of operations for block/transaction validation. This includes removing interactions between the PoW system and the storage engine to support this form of implementation.

**Thoughts:**

The proposed implementation of PoS prioritizes abstraction in order to clearly distinguish the new functionality from the existing. This results in a more maintainable and overall understandable architecture as each module/subsystem is delegated its own specific task. Further possible evolution of the PoS system is clearly defined in this implementation as the subsystem can be modified without heavily affecting other components.

**SAAM**

|  |  |
| --- | --- |
| Stakeholder | NFRs |
| Developers | Maintainability. Modifiability |
| Users | Security, Reliability, Performance |
| PoS participants | Security, Usability, Performance |

**Developers:**

**Maintainability:**  This refers to the ability of developers to easily troubleshoot the system, identify and fix issues and in general maintain the codebase efficiently

**Modifiability:** This refers to the ability to easily change certain components within the system

**Users:**

**Security:** This refers to the protection of user data, transactions and assets from unauthorized access, tampering or attack

**Reliability:** This refers to the dependability and availability of the Bitcoin network for users given the introduction PoS

**Performance:** This refers to the speed, efficiency and responsiveness of the Bitcoin network ins processing transactions

**PoS participants:**

**Security:** This refers to the protection of stakers assets from risks such as theft, fraud or malicious attacks

**Usability:** This refers to the ease of use and convenience of the PoS process for stakers

**Performance:** This refers to the efficiency and effectiveness of the PoS algorithm in generating rewards and processing transactions

**Option 1: modifying current files existing in Bitcoin Core**

**Maintainability and modifiability for Bitcoin Core developers.** This option for implementation results in minimal structural changes to the existing codebase, making it easier to maintain and evolve theoretically. However, it may also result in increased code complexity due to the addition of PoS-related logic to existing files. Potentially making it harder to maintain

**Security, reliability and performance for Bitcoin Core users**. This option for implementation may introduce risks related to the potential complexity of combining PoW and PoS logic in the same files. If not implemented correctly a majority of the users NFRs are at risk.

**Security, usability and performance for PoS participants.** This option may offer less flexibility in implementing the PoS algorithm and managing forgers due to the implementation being mixed in with the PoW algorithm.

Overall it can be seen that this option for implementing PoS into Bitcoin Core heavily prioritizes maintaining the structure of the architecture and as a result risks a lot of NFRs from prominent stakeholders. There is no prominent NFR that this form of implementation prioritizes.

**Option 2: creating new modules for PoS algorithm and forgers:**

**Maintainability and modifiability for Bitcoin Core developers.** This option may result in a cleaner separation between PoS and PoW logic making it easier to maintain and evolve. It may also result in easier identification and isolation of issues compared to option one due to this distinct separation

**Security, reliability and performance for Bitcoin Core users.** This option offers better isolation of PoS logic reducing the risks of potential code conflicts with the existing PoW algorithm. Although this implementation does not directly address these concerns, it does demonstrate less risk then the prior option

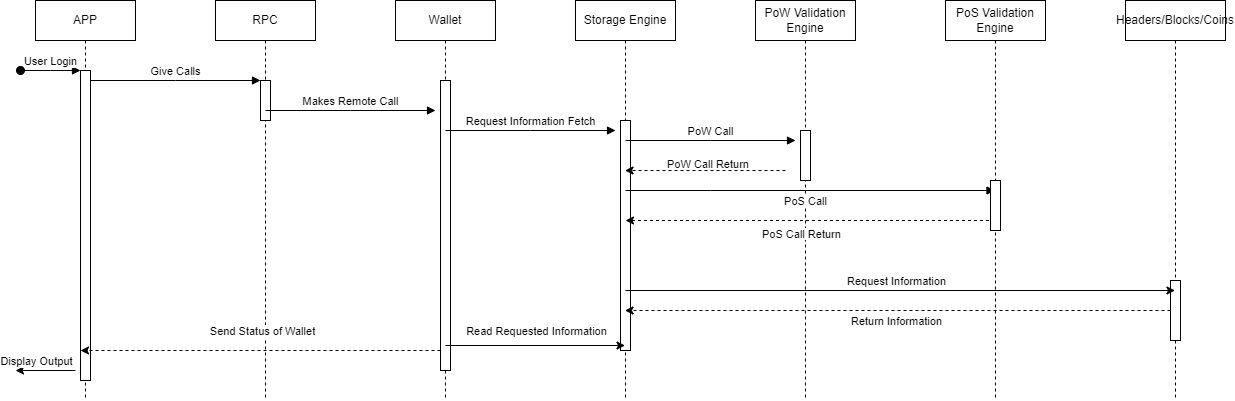
**Security, usability and performance for PoS participants**. This option offers better flexibility in terms of implementing the PoS algorithm as well as managing PoS participants in the long term. Once again this proposed implementation does not directly address the concerns but it does provide a better solution then option one.

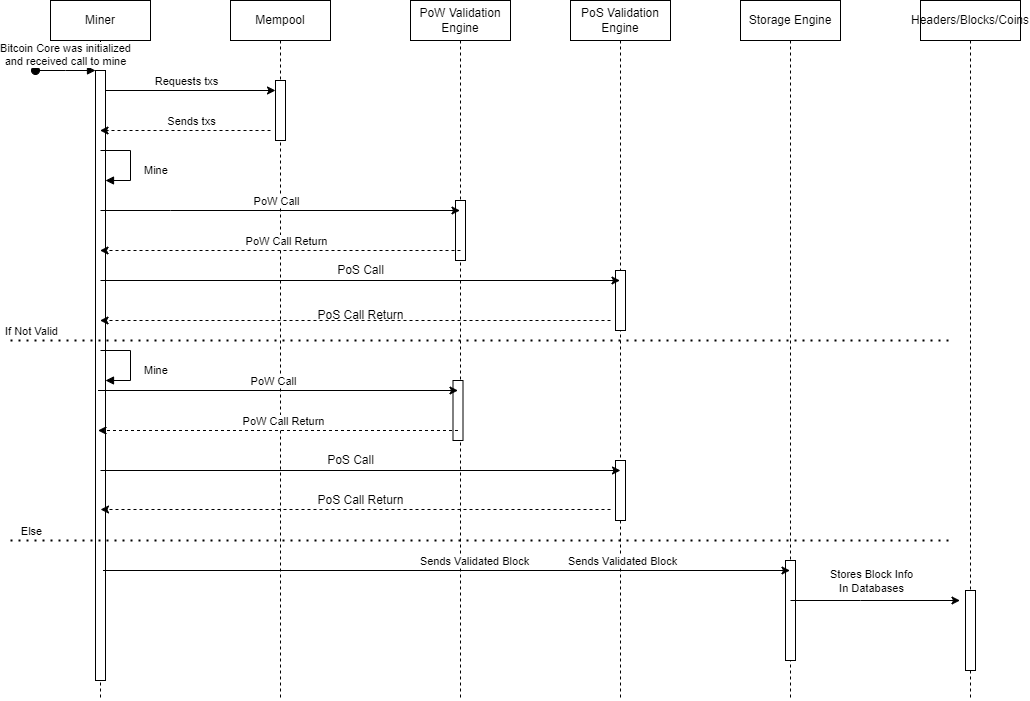
It can be seen that this implementation of PoS into Bitcoin Core follows a more practical approach by isolating the new functionality to specific modules. This approach prioritizes maintainability and modifiability and although it does not target other NFRs directly, it still performs better than the other presented option.

**Chosen Proposal through SAAM Analysis**

Out of the two proposed enhancements, the second enhancement would be the one that we would implement as it works better within the system and causes fewer disruptions. The first enhancement implementation is beneficial as it does not negatively impact or break the pre existing structure of the architecture, but is overall less versatile and applicable compared to the second implementation.

**Use Cases**



When a user wants to login, the App sends Util to check version and other utility functions, once that is returned, App sends the RPC a call that it wants the RPC to give the Wallet. The Wallet requests an information fetch from the Storage Engine, which fetches the requested information from the correct database. Prior to the fetch going through, two checks with the new PoS/PoW component must be performed. If both return valid, the use case can continue. The Storage Engine receives that information, which is read by the Wallet and sent back to the App to be displayed to the User.

When a user wants to mine and mint a new coin, the Miner receives a call to mine. The Mempool receives a txs request and returns it back to the Miner. The Miner then performs the mining action, until it needs validation on the created block. It requests validation from the Validation Engine and the validation status is returned. It also checks validation using our new component. If it's invalid, the process repeats and then validation is checked again. When it's valid, the block is sent to the Storage Engine. The Storage Engine sends the data to Utils, which compresses the data for more efficient storage in the databases. The Storage Engine receives the compressed data back and sends it to the appropriate database for storage.

**Plan for Testing**

To design practical tests for our proposed enhancement and its interaction with the Bitcoin Core architecture, we must create a set of tests that satisfy usability, privacy, testability, performance, and stability. We will create an individual test for both the proposed enhancements. The first will test for performance and stability, and the second test will test for useability and testability. The first test would be simulations as a means to monitor performance and stability. We would simulate both high and low network traffic and monitor how they both affect the system. This would provide us with conclusive results on the properties we were testing. The second test would be for the implementations usability and testability using performance testing. This test involves putting a very high volume of traffic through the system, with varying types and sizes. This will return results that will be indicative of both the performance of the system with our added component and also be a litmus test for the general testability of the updated program.

**Potential Risk**

The implementation of our enhancement might result in maintainability and evolvability risks. The increase of components and their necessary interactions with other components could result in a system much less malleable to future enhancements and changes. The enhancements proposed pose no security or useability risks to the current iteration of the system but as more enhancements are introduced, our proposed enhancements become more and more likely to cause problems in terms of both maintainability and evolvability. Our architectural enhancement could become a cascading failure point, where a small problem could spiral into a much larger issue for the system through its connections in the system.

**Naming Conventions**

SAAM (Software Architecture Analysis Method) is a method for Analyzing the Properties of Software Architectures.   
PoS is Proof of Stake  
PoW is Proof of Work

**Lessons Learned**Our group has learned the difficulties of adding to a generally complete system. Regardless, while attempting to enhance the architecture of Bitcoin Core, we learned what a SAAM model is, what its function is, and why it's important for the discussion of architectural enhancement. We also learned the difficulties with using the SAAM model along with its many beneficial features. Through the three completed components of this assignment, we were able to understand the entire Bitcoin Core system from a much more objective view. This was how we were able to offer potential enhancements to the complete system.

**Conclusions**

To conclude our analysis of the software architecture of Bitcoin Core, we proposed an enhancement to improve the security and validity of the transactions made. Our group aimed to implement an effective PoS/PoW algorithm that allowed for users to stake their individual bitcoins as collateral for their future transactions. The system would combine both algorithms in order to fully optimize the pre-existing consensus algorithm and allow for even more ensured decentralization. We compared and contrasted the two different implementations of our enhancement, and while both of them have advantages and disadvantages, our group came to the conclusion that the second implementation would be preferable. This is because it implements the PoS module and the Forgers module as unique, separate subsystems within Bitcoin Core, allowing for further modifiability and maintainability. Having the enhancement exist as a separate code block also allows any developer to more easily change and update the new subsystems implemented. This implementation best allows for our group to improve Bitcoin Core without sacrificing any security or functionality within the system. Our group is grateful to have discovered an effective way of improving upon Bitcoin Core through thorough analysis of the system and subsystems.

**References**<https://medium.com/@bhagvankommadi/saam-software-architecture-analysis-method-36864cd8ea94>

<https://resources.sei.cmu.edu/library/asset-view.cfm?assetid=29288>