A Discussion on Blockchain Software Quality Attribute Design and Tradeoffs

John M. Medellin(✉) and Mitchell A. Thornton

Darwin Deason Cyber Security Institute, Southern Methodist University, Dallas, TX 75275, USA
johnmedellin@verizon.net, mitch@lyle.smu.edu

Abstract. The blockchain design pattern has many variations and is a concept that will continue to lead many implementations in the years to come. New design and implementation patterns are frequently being announced and the choices available continue to expand. The design patterns imply tradeoffs which are reviewed.

We begin by describing the components of a blockchain; network nodes, blocks and consensus in a concept. We further elaborate on the key characteristics of the various design areas that are available adding emphasis to those used in private blockchains.

The individual components can be designed in different ways and imply tradeoffs between such quality attributes as performance and security or availability. We conclude with an initial tradeoff matrix that identifies the quality attributes that one should look for in designing these software systems.

Keywords: Blockchain · Cyber-attacks · Secure software architectures · Software architecture attributes · Software design tradeoffs

1 Introduction

Blockchain has become a common "house-hold" word in the vocabulary of almost all technology. This relatively new pattern of data sharing and encapsulated validation has impacted many public and private organizations. What started with simple exchange in values through crypto-currencies has become a veritable eco-system of different design and implementation choices. An experienced designer must still consider the far-reaching implications of selecting one course of action over another.

From its roots, deep in distributed operating systems and databases through current implementation, the pattern for information hiding through encryption and sharing of that information with other parties has continued and is continuing to evolve through requirements that are sometimes unknown until implementation. The pattern has been made famous by the implementation of bitcoin, a crypto currency. In a primary objective, the pattern allows for transferring value between the peers through consensus and encryption of results in a cumulative event ledger.

However, in addition to exchanging value concepts, the pattern is also very useful for sharing secrets or other information between participants. This particular sharing has been adopted by researchers and industry advocates for transmitting sensitive or

M. H. Miraz et al. (Eds.): iCETiC 2019, LNICST 285, pp. 19–28, 2019.
https://doi.org/10.1007/978-3-030-23943-5_2

private information between trusted parties in a group [1]. An example of this particular sharing is the transmission of PKI (private key infrastructure) keys in a network where only the intended receiver can decode such a transmission and by virtue of the blockchain pattern cannot repudiate (negate receipt) of that specific information. Furthermore, a block is mathematically bound with subsequent blocks through progressive hashing. The cumulative effect of this is to make the transaction a permanent one in the event registry.

This objective of this paper is to segment the key software quality attributes and associated tradeoffs a designer should consider when architecting a solution with this pattern in mind. As mentioned above, the standard is evolving and will continue to do so for the foreseeable future. We discuss our assessment of these key tradeoff patterns and provide a matrix that identifies the ones that should be especially considered in software design. We however do remind the reader that these are only considerations not to be omitted in addition to consideration of all software quality attributes for any solution design.

2 Related Work

Many variations have been had to the famous Nakamoto [2] paper; the precursor to the famous bitcoin series of crypto currencies. This first paper on the nature of exchange of value through the blockchain design pattern has been adopted and now constitutes several billion currency units of value in over 700 crypto exchanges around the world; bitcoin being the most famous.

Since Nakamoto in 2008, most of the implementations of blockchain have had crypto currency as their target. However, in recent studies, the design pattern has begun to be adopted as a means to transmit data while preserving the key attributes of privacy and encryption. Indeed, many technology implementations have begun to look at this design pattern as a means to protect private exchange of information between interested parties [3, 4].

A significant difference in implementation patterns exists when a private blockchain is implemented versus a public blockchain [5]. In a public blockchain environment, parties do not necessarily have to know or trust each other in order to exchange value. Rather they can exchange by creating a block that is cryptographically validated by others in the network known as miners who are incented to validate by receiving shares of crypto currency for the one that validates the block first.

In contrast in a private exchange of data, the parties wish to know and trust each other before the exchange happens. Some examples of private blockchains include distribution of sign-on credentials [1] or authentication of agents delivering content on a home [6]. In these cases, the “value” to be exchanged is a secret or information that is only known to the sender and receiver [7] and potentially to other trusted parties in the closed network of operation for the blockchain.

As previously mentioned, significant amounts of intellectual capital have been spent on public blockchains. However, our focus is the private blockchain as a means to guarantee key properties of secure transmission models. We have written about

usage of this design patterns in previous work and have focused mostly on the consensus architecture requirements and implementation [8].

Our previous work was dedicated to modeling different consensus algorithms and their impact on resource consumption. Most of the literature compares those algorithms to the Byzantine General's Problem [9] since it has become a pseudo-standard for measuring performance characteristics in newer algorithms. In this work however, we expand the scope of our discussion to include the other aspects of the blockchain architecture requirements in relation to Private exchanges. In a later section we discuss the block, smart contract, network, consensus and cryptography aspects.

3 Design Considerations

There is some variability in the definition of components required for a successful implementation. However, there is general agreement that the components required for the pattern to work include:

- A block architecture; the contents and specifications of the actual block of data to be transmitted [10].
- The scope and specifications of the smart contract; a smart contract is a reference to a set of procedures that the block will operate or has been operated in an offline fashion [11].
- The consensus model; the process whereby the participants agree in adding the new block to the chain [12].
- The network of participants; the group of valid members that can perform the basic operations of exchange and validation [13].
- The cryptography algorithm; the method for encrypting the block by carrying forward necessary information, a mathematical formula result and a unique number a "nonce" or number that is only used once [14].

3.1 General Architecture Requirements by Component

A brief description of each component was previewed above, this section discusses them in more detail.

1. Block: this is the specific unit of information that is "chained" by reference to the prior block. The block must be able to contain the following components (Fig. 1 below from ethereum.org):
   a. A mathematical reference to the prior block, usually in a part of the encryption result from that prior block.
   b. A mathematical reference to a unique number included as input into the block encryption algorithm (the "nonce").
   c. A mathematical reference to the actual data being encrypted.

The result of the encryption algorithm (for example RSA, or SHA-512 or SHA-256 [15]) as required by the overall parameters of the implementation and combining the three previous components.

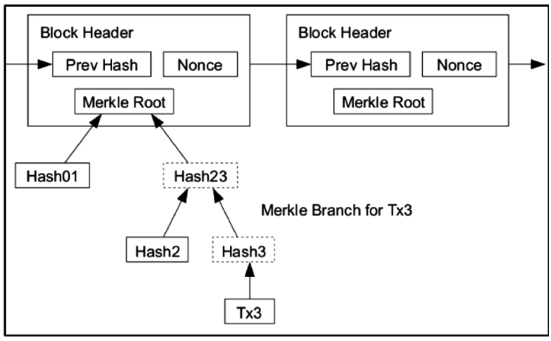

Fig. 1. Blockchain architecture.

2. Smart Contract: this is an optional item but one which is quite popular in business transaction processing. The smart contract is a block that defines a set of procedures in the actual data payload or a reference to another part of the blockchain with those references. The smart contract, when invoked by a special block-type, will produce a result that operates on the input data of that other block. The smart contract may contain the algorithm for processing the additional data or may reference a specific location for those instructions [16].
3. Network: Network in this context means the group of members that are on-boarded, validated, able to operate and off-boarded in the blockchain [17].
4. Consensus Model: This is the method whereby sufficient proof of validity is provided by a given set of participants in order to append the block and commit it to permanent storage in the chain. The two most popular ones appear to be Proof of Work (PoW) [18] in public blockchains and Byzantine Fault Tolerance Consensus (BFTC) [19] in private. Discussion of these details is beyond this paper but in a succinct explanation is that PoW deals with presentment of sufficient available computation usage to validate the block while BFTC deals with validation of voting by valid voter nodes [13].
5. Encryption Method: This pertains to the mathematical technique used to encapsulate the block into a number that is computationally inefficient to derive before the next block is appended to the chain [20].

Next, we will describe each one of the above components in light of variations that can be included in the implementation of those blockchains.

3.2 Key Variability Parameters by Architecture Component

Each of the architecture patterns has a variety of choices that ultimately influence the achievement of objectives in the blockchain implementation. The major component variabilities and impacts are discussed below.

1. Block: the block architecture itself is a major influencer on the objectives. On one hand, the block needs to be sized appropriately in order to include the necessary data. On the other however, if the block is too large the impact is felt in encryption

computation and latency of transmission [18]. Still on the other hand, a large block provides by definition a larger cyber-attack surface.

2. Smart Contract: The nature of the smart contract has a very large impact on both performance and attack resilience. By including a smart contract in the implementation complexity and additional overhead is acquired [13]:
   a. If the contract is wholly contained in the design then by definition, some of the prior blocks will need to be modified. A wholly contained smart contract is a type of block that is mutable as it processes and receives the effect of transactions.
   b. If the contract has a reference to a third location, the attack surface is expanded, performance could be impacted by additional steps to be taken in retrieval and update of external entities and finally, for subsequent block encryption, this result must be taken into consideration so the cryptography rules can continue unimpaired.
3. Network: the variability in the network aspect is the rules for adding/processing/deleting nodes/actors in the chain. This might seem to be the most secure but it is where Impostor and Sybil [7] attacks have most frequently come from. The variability in private blockchain comes as a function of the complexity to join versus the usability of the method. A strict adherence to high trust necessitates high complexity to belong in most cases. That additional overhead needs to be balanced with the usability of the blockchain.
4. Consensus Model: this is the area of highest variability in the implementation design. There are several dozen consensus models that have both computational overhead and complexity required to ensure agreement without tampering. Several discussions on these algorithms are out there [21, 22].
5. Encryption Method: most of the blockchain patterns defined will use accepted methods of cryptography and these are well documented [15]. The biggest issue identified has been the potential for quantum computing usage to break those algorithms. Several studies exist on post-quantum cryptography and we strongly suggest incorporating some of those initial rules into the design [23].

3.3 Software Architecture Quality Attributes

The Software Engineering Institute (SEI) has published guidelines for designing software systems with emphasis on architecture [24]. They are as follows:

1. Availability; the system is there and ready for interaction.
2. Modifiability; the system is able to be altered.
3. Testability; includes facilities for validation of results.
4. Interoperability; able to interact with other systems.
5. Security; resilience to cyber-attack.
6. Performance; achieves SLA targets.
7. Usability; user able to use system for designated purpose.
8. (Optional) Extensibility; able to extend functionality beyond its original enterprise level scope.

These attributes are key in determining the quality of the software system and usually work against each other (for example, performance vs security). Our main contribution in this paper is the focus of our next section; how do these tradeoffs work to enhance or deter each other.

4 Quality Attribute Tradeoffs

Software Quality Attributes will drive decisions that will shape the final solution [25]. In this section, we discuss the overall tradeoff process while also elaborating on the tradeoffs that are more prevalent in the blockchain design pattern.

4.1 Architecture Design Implies Tradeoffs

Architecture design implies tradeoffs between quality attributes in order to deliver a solution that reasonably complies with stated requirements. Quality attributes are typically in conflict with each other, for example, the testability attribute which includes verbose output might be in conflict with the security attribute which aims at hiding the attack surface (verbose output implies additional code that may be turned on to describe what's happening but this code could also be attacked). Another example would be the conflict between interoperability and performance. If a system is designed with very high level of interface abstraction and cohesion it might cause the system to use more cycles and impact performance requirements. These tradeoffs are even more specific for blockchain design and are discussed in the next section.

4.2 Key Blockchain Architecture Tradeoffs

These are some of the more important tradeoffs in blockchain architecture design. They are complemented further from the list provided by [26–28]:

1. Storage vs Computation: the amount of storage required by the block will affect the computation requirements; a larger block will require more computation to encrypt since there are more elements to encrypt. The design tradeoff in this case is *Usability vs Performance*.
2. Anonymity vs Trust: if the parties do not know each other they most likely need to verify proper identities through the use of public/private keys. This additional step will require a different design pattern. The design tradeoff in this case is *Interoperability vs Security*.
3. Incentive variations: by definition, the pattern requires distributed validation before blocks can get appended. Participants are usually incented to validate in order to carry out this function. In public chains, this means allocation of value to those participants which implies tracking value in the network. Similarly, in private, the incentive schemes vary, in the simplest form it could mean that the participant must validate prior blocks before theirs gets appended. The design tradeoff in this case is *Extensibility vs Usability*.

4. Degree of Distribution: In traditional Nakamoto patterns, the participants keep copies of the blockchain. However, there are different patterns which may require that smart contracts reside offline, on the chain in a centralized or other node. The design tradeoff in this case is *Interoperability vs Extensibility*.
5. Scalability vs Latency: These two attributes are closely tied into performance. In most cases, systems are designed for fulfilling a certain return time to users that is specified in requirements. When those limits are exceeded by larger scale (more volumes) the processing of blocks begins to lag, in some cases well beyond the expected requirements. The design tradeoff in this case is *Availability vs Performance*.
6. Immutability vs Process Functionality: This happens mainly on implementation of smart contracts. One of the key tenants of the architecture is that it is a permanent record. The inclusion of smart contracts can either violate (by having data offline or modifying previous blocks) or preserving it (by restating the smart contract, previous states and the of new state). The design tradeoff in this case is *Security vs Usability*.
7. Consensus Algorithm Selection: This tradeoff relates to the selection of the consensus approach to validation. In Nakamoto for example, the PoW constitutes both a security approach (51% of the omputation) and validation of the block (encryption/finding the "nonce") while in other approaches, this could be varied to provide a level of accuracy and protection that may not be as computationally intense. The design tradeoff in this case is *Security vs Performance*.

4.3 Architecture Tradeoff Matrix

We next proceed to defining the tradeoff attribute matrix. The numbers in cells refer to the discussion number in 4.2 above.

	Availability	Interoperability	Modifiability	Performance	Security	Testability	Usability	Extensibility
Availability	X			5				
Interoperability		X			2			4
Modifiability			X					
Performance	5			X	7		1	
Security		2		7	X		6	
Testability						X		
Usability				1	6		X	3
Extensibility		4					3	X

Fig. 2. Key architecture quality attribute tradeoff matrix

5 Tradeoff Discussion

Bearing in mind that a holistic view of all quality attributes is best practice [24] one should design for all quality attributes (by usage of patterns that facilitate them and embed in design [29]). However, some quality attributes are key in different architectures and will typically require additional tradeoff analysis to solve [30]. Figure 2 above depicts the major quality attribute tradeoffs identified in earlier sections of this document.

5.1 Key Attribute Tradeoff Discussion

From a purely numeric perspective, it is evident that performance, security and usability are the most impacted attributes. This intuitively makes sense since performance is taxed by adding requirements for security and usability (both of which require additional resources and potentially more design impacting performance decisions). Interoperability and extensibility are a close second in this tradeoff analysis. This also intuitively makes sense since blocks are dependent on each other to deliver functionality and the variety of design patterns requires extensibility to enable design those additional features. Finally, availability (although only one tradeoff is noted) is a very important quality attribute since if the blockchain system is unavailable or underperforms (as in the case of some crypto currencies) it will render the system unusable.

The modifiability and testability attributes do not seem to have specific tradeoffs (beyond those normal software systems require). This also intuitively makes sense since to our knowledge there are no specific requirements to modify code or test that code beyond what would be required in normal transactional or other types of systems.

5.2 Use of the Key Attribute Tradeoff Matrix

As mentioned above, all attributes should be considered in the design decisions (most design patterns incorporate them [31]). The objectives of this document are to convey which quality attributes seem to be more important from our research. The matrix should be used as an additional checkpoint to ensure these key issues are considered by the designer of blockchain systems.

References

1. Li, D., Du, R., Fu, Y., Au, M.H.: Meta-key: a secure data-sharing protocol under blockchain-based decentralized storage architecture. IEEE Netw. Lett. **1**(1), 30–33 (2019)
2. Nakamoto, S.: Bitcoin: A Peer-to-Peer Electronic Cash System. www.bitcoin.org. Accessed 19 Mar 2019
3. Dinh, T.T.A., Wang, J., Chen, G., Liu, R., Ooi, B.C., Tan, K.-L.: BLOCKBENCH: A Framework for Analyzing Private Blockchains. https://arxiv.org/pdf/1703.04057.pdf. Accessed 19 Mar 2019

4. Dorri, A., Kanhere, S.S., Jurdak, R.: Towards an optimized blockchain for IoT. In: 2017 IEEE/ACM Second International Conference on Internet-of-Things Design and Implementation (IoTDI), pp. 173–178 (2017)
5. Medellin, J., Thornton, M.: Simulating resource consumption in three blockchain consensus algorithms. In: MSV 2017 International Conference on Modeling, Simulation & Visualization Methods, pp. 21–27 (2017)
6. Dorri, A., Kanhere, S.S., Jurdak, R., Gauravaram, P.: Blockchain for IoT security and privacy: the case study of a smart home. In: 2017 IEEE International Conference on Pervasive Computing and Communications Workshops (PerCom Workshops), pp. 618–623 (2017)
7. Salman, T., Zolanvari, M., Erbad, A., Jain, R., Samaka, M.: Security services using blockchains: a state of the art survey. IEEE Commun. Surv. Tutorials **21**(1), 858–880 (2019)
8. Medellin, J., Thornton, M.: Performance characteristics of two blockchain consensus algorithms in a VMWare hypervisor. In: 2018 International Conference on Grid & Cloud Computing and Applications "GCA 2018", pp. 10–17 (2018)
9. Lamport, L., Shostak, R., Pease, M.: The byzantine generals problem. ACM Trans. Program. Lang. Syst. **4**(3), 382–401 (1982)
10. Liang, X., Wu, T.: Exploration and practice of inter-bank application based on blockchain. In: The 12th International Conference on Computer Science & Education (ICCSE 2017), pp. 219–224 (2017)
11. Christidis, K., Devetsikiotis, M.: Blockchains and smart contracts for the internet of things. IEEE Access **4**, 2292–2303 (2016)
12. Ongaro, D., Ousterhout, J.: In search of an understandable consensus algorithm. In: Proceedings ATC 2014 USENIX Annual Technical Conference. USENIX (2014)
13. Muralidharan, S., Murthy, C., Nguyen, B., et al.: Hyperledger Fabric: A Distributed Operating System for Permissioned Blockchains. https://arxiv.org/pdf/1801.10228.pdf. Accessed 19 Mar 2019
14. Ferguson, N., Schneier, B., Kohno, T.: Cryptography Engineering Design Principles and Practical Applications. Wiley Publishing Inc, Indianapolis (2010)
15. Stallings, W.: Cryptography and Network Security, Principles and Practice, 7th edn. Pearson Education Limited, London (2018)
16. Daniel, F., Guida, L.: A service-oriented perspective on blockchain smart contracts. IEEE Internet Comput. **23**(1), 46–53 (2019)
17. Xia, Q., Sifah, E.B., Asamoah, K.O., Gao, J., Du, X., Guizani, M.: MeDShare: trust-less medical data sharing among cloud service providers via blockchain. IEEE Access **5**, 14757–14767 (2017)
18. Fullmer, D., Morse, A.S.: Analysis of difficulty control in bitcoin and proof-of-work blockchains. In: 2018 IEEE Conference on Decision and Control (CDC), pp. 5988–5992 (2018)
19. Golosova, J., Romanovs, A.: The advantages and disadvantages of the blockchain technology. In: 2018 IEEE 6th Workshop on Advances in Information, Electronic and Electrical Engineering (AIEEE), pp. 1–6 (2018)
20. Johnsonbaugh, R.: Discrete Mathematics, 8th edn. Pearson Education Inc, New York (2018)
21. Ehmke, C., Wessling, F., Friedrich, C.M.: Proof of property – a lightweight and scalable blockchain protocol. In: 2018 IEEE/ACM 1st International Workshop on Emerging Trends in Software Engineering for Blockchain (WETSEB), pp. 48–51 (2018)
22. Ceccetti, E., et al.: Solidus: Confidential Distributed Ledger Transactions via PVORM CCS 2017, pp. 1–23, 30 October– 3 November 2017
23. Li, C.-Y., Chen, X.-B., Chen, Y.-L., Hou, Y.-Y., Li, J.: A new lattice-based signature scheme in post-quantum blockchain network. IEEE Access **7**, 2026–2033 (2019)

24. Bass, L., Clements, P., Kazman, R.: Software Architecture in Practice: The SEI Series in Software Engineering, 3rd edn. Addison Wesley, Upper Saddle River (2012)
25. Cervantes, H., Kazman, R.: Designing Software Architectures; A Practical Approach: The SEI Series in Software Engineering. Pearson Education, Boston (2016)
26. Scriber, B.A.: A framework for determining blockchain applicability. IEEE Softw. **35**(4), 70–77 (2018)
27. Xu, X., Weber, I., Staples, M., Zhu, L., et al.: A taxonomy of blockchain-based systems for architecture design. In: Proceedings 2017 IEEE International Conference on Software Architecture, pp. 243–252 (2017)
28. Zheng, Z., Xie, S., Dai, H., Chen, X., Wang, H.: An overview of blockchain technology: architecture, consensus, and future trends. In: 2017 IEEE International Congress on Big Data (BigData Congress), pp. 557–564 (2017)
29. Booch, G.: Object-Oriented Analysis and Design with Applications. Addison Wesley Longman, Inc., Reading (1994)
30. Tian, J.: Software Quality Engineering; Testing, Quality Assurance and Quantifiable Improvement. IEEE Computer Society (2005)
31. Larman, C.: Applying UML and Patterns: An Introduction to Object-Oriented Analysis and Design and Iterative Development, 3rd edn. Pearson Education Inc, Upper River (2005)