Byzantine Fault Tolerance

Stellar Consensus Protocol

Group Members
Vivek Aggarwal (**va713**), Shengchi Chang (**sc4889**), Omkar Keluskar (**ork216**)
Manish Nagdevani (**man514**), and Vivian Zhao (**vz341**)

December 7, 2017

1. Abstract

This paper is an attempt to solve the certificate or certification authority (CA) accountability problem. In a decentralized CA system, the biggest concern that raises CA accountability issues is the Byzantine agreement problem in voting systems. The proposed solution addresses this problem by applying Stellar consensus protocol (SCP). All the CAs form a quorum and vote using the ballot-based approach to reach a consensus on a statement. The proposed solution guarantees no indefinite blocking of the system due to any faulty or malicious nodes, and tries to minimize the blocking stages by neutralizing the blocking statements.

2. Introduction (with Motivation)

A CA is a third party entity that issues or revokes digital certificates. A digital certificate validates the ownership of a public key by the certificate owner, which enables relying participants to depend upon signatures or on assertions made about the private key that corresponds to the certified public key. A distributed certificate or certification authority (DCA) allocates the CA's private key to shareholding DCA nodes, but the DCA's public key will be known by all network's nodes and used to verify certificate signatures issued by the DCA.

Often times, CAs sign certificates are utilized in HTTPS, the secure browsing protocol for the World Wide Web (WWW), and issue identity cards by national governments utilized in electronically signed documents. An implicit yet significant trust is placed on CAs, but are they accountable? This is especially troubling if the CA system is decentralized, which means any participant is allowed and no central authority controls whose approval is required for consensus. In addition, it is known that CAs sign incorrect certificates that are publicly utilized. Several proposals regarding certificate transparency address this issue. Certificate transparency enables users to inspect the history of certificates issued for any given entity, and detect attempts by CAs to modify an entity's public key without the endorsement of the previous key.

This paper presents a model that is suitable for worldwide consensus called federated Byzantine agreement (FBA). In FBA, each participant is cognizant of others it considers to be important. Each participant waits for the vast majority of others to agree on any certification. Similarly, each participant considered to be important does not agree to the certification until the participants they consider to be important agree as well, and so forth. Eventually, a sufficient

amount of the network accepts a certification that it becomes infeasible for an attacker to roll it back. Only then do any and all participants consider the certification to be settled. Therefore, FBA's consensus can ensure the integrity of CA.

This paper also presents a construction for FBA called SCP. SCP's safety is optimal for an asynchronous protocol, which guarantees agreement under any node-failure scenario that admits to such a guarantee. It is without blocked states or when consensus is no longer possible, unless participant failures make it impossible to meet trust dependencies.

The possibility to strengthen the ineradicable certificate history lies within SCP. Demanding global consensus on certificate history among a decentralized group of inspectors would make it difficult to reverse and override previously issued certificates.

The next section defines FBA and its applicable notions of safety and liveness. Section 4 discusses threat models of the use of SCP in the decentralized CA accountability problem. Section 5 develops the design of the proposed solution. Section 6 presents the implementation of the proposed solution. Section 7 evaluates the proposed solution. Section 8 provides related solutions. Section 9 concludes this paper. Lastly, references can be found in Section 10.

3. Background

The problem to be presented is concentrated on the issuing of certificates by CA. With the current system, if any of the root CAs is hacked, then the hacker can easily issue multiple certificates for any website that will be seen as legitimate, enabling man-in-the-middle attacks without detection. Therefore, a system that is recognized by all CAs must reach a consensus to issue a certificate to a particular website is proposed. To fully comprehend this proposed solution, some key concepts must be understood as well.

In distributed systems, Byzantine fault tolerance (BFT) is when the system is tolerant of a fault called the Byzantine general's problem. In these kinds of systems, there are multiple nodes that communicate with each other, so that they can all reach a consensus on a single output. However, in the Byzantine general's problem, one or more of the nodes might be "treacherous" and send-out different messages to different nodes, which can have a drastic effect on the final consensus reached by the nodes. In the proposed solution, the system must be Byzantine fault tolerant, so that the protocol can complete without communicating to a specific node. The goal is to minimize the effect that such "treacherous" nodes will have on the final output.

In FBA, the set number of nodes and how trustworthy each node is are unknown. Therefore, the quorums are determined in a decentralized way such that each node has the ability to choose quorum slices, which are nodes that a particular node determines to be important, and if every node in the quorum slice arrives to a single consensus, then that node must vote for that specific consensus as well.

SCP is a general FBA protocol that can guarantee the safety for well-behaved nodes that have quorum intersection, and that intact nodes will not get stuck. However, if there are too many ill-behaved nodes, then the result can still be diverged. The main idea of SCP is based on federated voting. If a set is claimed to be v-blocking, it means that either the entire set is false, or

the statement claimed by the set is true. To solve the issue that not all intact nodes can accept a statement, and that there is suboptimal safety for non-intact nodes in a quorum intersection, a second vote is held after the first one. This also brings back the theorem that once a single intact node confirms a statement, all other intact nodes will eventually agree with the statement as well.

To bring these concepts back to the presented problem, there are currently multiple root CAs such that any of them can issue legitimate certificates for any website. To solve this problem, a modified version of the SCP will be applied, so that all CAs would need to reach a consensus before issuing a certificate for any website. SCP is the key to strengthening a certificate's history and future certificate transparency. The main goal is to maximize the safety, liveness, and fault tolerance of the system.

4. Threat Model

Currently, any trusted CA can issue a certificate for any website. However, since there are many CAs worldwide, governments can pressure the CAs active in their nation to issue a SSL certificate for any site they desire to impersonate. For example, it was discovered that a French CA had issued a rogue certificate for google.com, which allowed man-in-the-middle attacks to occur, compromising authentication, confidentiality, and integrity of information.

Another threat is that if any single CA is compromised, it puts almost any website at risk, because any trusted CA can issue certificates for any website. To minimize this risk, there should be a requirement for all CAs to reach a consensus before issuing a certificate to any website.

One vulnerability that exists in FBA is that each node is allowed to choose its quorum slice. For example, some CAs might have a better reputation for their security than other CAs that might have been breached before. Therefore, these more "influential" CAs would most likely exist in more quorum slices than the other CAs, providing them more influence over the final result that the protocol will output.

A threat to FBA is that if one of these more influential CAs is compromised, then the outcome of the ballots could be vastly influenced, which makes these reputable CAs more at risk for attacks. Currently, any CAs have the same "influence" in the sense that any certificates that they issue would be considered legitimate. This allows attackers to attack the weakest CA and gain the same amount of access and influence as hacking into a bigger and more reputable CA.

5. Design

The proposed solution to the Byzantine agreement problem in decentralized CA systems uses SCP. SCP guarantees safety, liveness, and fault tolerance in distributed systems thus facilitating the system to reach all the goals described in this paper.

Let all CAs be the nodes of the system. Let there be N such nodes. Quorum (T) will be the majority of the N nodes, which is $(N/2) + 1$, therefore, $T > N/2$. For the sake of simplicity, let there be two types of statements S_1 = "Issue Certificate" and S_2 = "Do not Issue Certificate." In order to reach a consensus on S_1 or S_2, at least T nodes must vote in the favor of either S_1 or S_2.

Each node in the system forms its quorum slice. A quorum slice is a group of nodes on which a node depends upon to make its decision on a statement. CAs can be chosen as a part of a quorum slice based on various factors such as trust, reliability, etc. A good quorum shares nodes and leads to quorum overlap called a quorum intersection.

When a website requests for a certificate to the system, each node in the system votes for either S_1 or S_2 based on the deterministic computation, such as authentication of the identity of the business, etc. This preliminary, individual voting by a node means that it agrees on a statement.

It is possible that the system might get blocked with no consensus. To solve this problem, the system must be able to neutralize the blocked statement. This is achieved by using a ballot-based voting approach. Ballot-based voting means that, to eventually externalize a statement, a node must commit the ballot tied to that statement.

Each node may either commit or abort a ballot. In this case, if the system is stuck with no agreement on a statement, then there is a need to neutralize that blocking statement. If the system gets stuck on S_1, then the system must accept to abort S_1, and the group moves on to vote for S_2.

If each node in the quorum slice vote the same on a statement, then it is called a ratification, in such a case, the node which depends upon this slice also makes its vote in favor of the ratified statement. If the system ratifies the statement, S_1, the system reaches a consensus, and the ballot tied to S_1 is confirmed by another round of voting to confirm the acceptance of S_1.

6. Implementation

A mock implementation was created in JavaScript using Node.js, JSON Web Tokens, and sockets. Its link to the GitHub repository is referenced in Section 10. This serves as a good visual representation of SCP over the network. An advantage of using Node.js would be the non-blocking IO.

Quorum: T nodes in the system form a quorum from N nodes, therefore $T > (N/2) + 1$. Quorum votes on a new nomination, in this case just one, whether to allow the CA to sign a certificate. If T nodes in the system vote for the nomination, for every ballot slot, then the certificate is issued. Nodes that form quorum splice vote depending upon the decision being made in that splice. So for instance, the personal node forms a quorum splice with NYU, CUNY, and Columbia. If NYU and Columbia vote for issuance of certificate, then for that splice, majority is “Issue Certificate.” Henceforth, CUNY would agree and so would the personal node. This way, nodes reach a consensus.

Nodes: Newer nodes can join the quorum and form their own quorum splice. Trusting newer nodes can be tricky, especially for bad actors. So, an implementation of trust over reputation of CA (NYU over the personal node) was done. Trusting newer nodes means opening problems in SCP, but trying each node before trusting them is preferable.

Voting: Each node in a quorum sends an initial nomination for the vote. So if majority vote is either “Issue Certificate” or “Do not Issue,” accordingly the ratification is done. But there might be some Byzantine timeouts or network conditions, causing the system to halt and affect

liveliness. In that case, the system would try to neutralize the blocked statement, and eventually externalize the statement.

Byzantine Timeouts and Faults: The protocol would work as intended as the nodes just have to trust their neighbors in the quorum splice. So even if there are some bad actors, the system would survive, or a node can just change its trust from a quorum splice to another, to try out other nodes and build a wider trust over the network.

Working: A mock environment is built using sockets to communicate between nodes. Some REST APIs create or change quorum splice out of existing nodes in the quorum. For a given event, to not issue a certificate, each node in the quorum must vote with a "yes" or "no" message. Nodes in a splice vote when majority of the nodes in their splice agree on a message. This way a consensus is reached, whether or not to issue a certificate.

7. Evaluation

Using SCP for CA accountability problem helps to solve the transparency of certificates being issued by CAs as well as issuance of new certificates.

Making use of FBA to decide on whether or not to issue certificates lays out notions of safety and liveness to SCP.

In past, there have been several cases where an intermediate CA would issue a certificate for a reputed domain, but it was used by attackers to perform MiTM, spoofing attacks. One such case was when an intermediate CA, CNNINC, issued an unauthorized digital certificate for *.google.com domains. CNNINC is a trusted source in most browsers and OS. So any misuse of such a certificate would have remained undetected.

The proposed implementation helps to solve that problem. Using SCP to reach a consensus by forming quorums and to not issue the certificates within acceptable time delays. Not only does this help to prevent unauthorized issuance of certificates, but also promotes certificate transparency. As taking quorum consensus on history of certificates in a decentralized network would make it harder to repudiate and override previous certificates.

8. Related Solutions

The most well-known, decentralized consensus mechanism is the proof-of-work scheme. However, it wastes resources, and secure settlement suffers from expected latencies. In a stark contrast to traditional cryptographic protocols, proof-of-work does not offer asymptotic security. For example, if there are non-rational attackers or those with ulterior motives to sabotage consensus, minor computational advantages can disprove the security assumption, which enables history to be rewritten in so-called "51% attacks." In the worst case scenario, attackers originally controlling less than 50% of computation can cheat the system to provide disproportionate rewards for those who align with them. Therefore, they may potentially gain majority control. However, smaller systems were targeted, which poses a problem for any proof-of-work system not developed on a strong blockchain.

An alternative to proof-of-work is proof-of-stake, which has consensus depend upon participants who offered collateral. Similar to proof-of-work, incentives encourage rational participants to follow the protocol. In addition, some designs penalize poor behavior. However, proof-of-stake enables the possibility of so-called "nothing at stake" attacks, which is when participants who previously offered collateral, but used it later can revisit and rewrite a point in time when they still had stake. To mitigate such attacks, systems effectively combine proof-of-stake with proof-of-work, which decreases the required work in proportion to stake, or delay compensating collateral for a sufficient amount of time so that another consensus mechanism can establish an irreversible checkpoint.

Another solution to consensus is Byzantine agreement, and the best-known variant is practical Byzantine fault tolerance (PBFT). Byzantine agreement ensures consensus despite arbitrary, which includes non-rational behavior on the part of a minor amount of participants. It enables consensus to be efficient, and trust to be completely decoupled from resource ownership, which makes it possible for a minor non-profit to aid in the maintenance of more powerful organizations, such as CAs to be trustworthy. However, all participants are required to agree upon the accurate list of participants. In addition, attackers are required to be barred from aligning many times and exceeding the system's failure tolerance in so-called "Sybil attack." Byzantine fault tolerance-consensus with unknown participants (BFT-CUP) accommodates unknown participants, but still presupposes a Sybil-proof centralized admission-control mechanism. SCP is the first Byzantine agreement protocol to provide each participant maximum freedom in deciding which combinations of other participants to trust.

9. Conclusion

A successful implementation of a decentralized and safe mechanism to issue SSL certificates approved by CA uses SCP. SCP is the first provably safe consensus mechanism to offer four key properties at the same time: decentralized control, low latency, flexible trust, and asymptotic security, which is shown in the Sections 5, 6, and 7. Therefore, the Byzantine fault in the decentralization of CA to issue SSL certificates can be solved by the proposed solution described in the paper.

10. References

Ahmadi, Mohammad R., Jamshid Bagherzadeh, Seyyed M. Hashemi, Sam Jabbehdari, Ahmad Khadem-Zadeh and Mohammad Masdari. "A survey and taxonomy of distributed certificate authorities in mobile ad hoc networks." *EURASIP Journal on Wireless Communications and Networking*, 27 Sept. 2011. *SpringerOpen*, doi.org/10.1186/1687-1499-2011-112.

"Certificate authority." *Wikipedia*, en.wikipedia.org/wiki/Certificate_authority. Accessed 22 November 2017.

"Fraudulent Digital Certificates Could Allow Spoofing." *Microsoft*, 3 Jan. 2013, docs.microsoft.com/en-us/security-updates/SecurityAdvisories/2013/2798897. Accessed 14 Jan. 2013.

Holliman, John, and Jeremy Rubin. “Oort: Stellar Consensus Protocol Implementation.” 8 May 2015, rubin.io/public/pdfs/oort.pdf.

Hoffman, Chris. “5 Serious Problems with HTTPS and SSL Security on the Web.” *How-To Geek*, 13 Feb. 2014, howtogeek.com/182425/5-serious-problems-with-https-and-ssl-security-on-the-web/.

Keluskar, Omkar. “Stellar Consensus for approving SSL certificates through Certificates authorities.” *GitHub*, 5 Dec. 2017, github.com/orkeluskar/scp-ca.

Langley, Adam. “Maintaining digital certificate security.” *Google*, 23 Mar. 2015, security.googleblog.com/2015/03/maintaining-digital-certificate-security.html.

Mazières, David. “David Mazières: ‘The Stellar Consensus Protocol’ | Talks at Google.” *YouTube*, uploaded by Talks at Google, 14 June 2016, youtube.com/watch?v=vmwnhZmEZjc.

Mazières, David. “The Stellar Consensus Protocol: A Federated Model for Internet-level Consensus.” *Stellar Development Foundation*. *Stellar*, stellar.org/papers/stellar-consensus-protocol.pdf.