Confidential Cloud Services

Neele Peter
neele.peter@fau.de
Friedrich-Alexander-Universität Erlangen-Nürnberg

Abstract

Nowadays, many services are being outsourced to the cloud, because of achieving more flexibility and scalability. The problem is that cloud services are very unsecure snd therefore not usable for applications with high security requirements. One solution is using Trusted Execution Environments (TEEs) to provide more safety. Therefore many services exist that integrated TEEs and other mechanisms to make the cloud environment confidential. These systems are called Confidential Cloud Services (CCS).
In this paper I compare two of these services, the Confidential Consortium Framework (CCF) and Nimble. They both fulfill the requirements of the CIA triad which are confidentiality, integrity and high availability. Especially when it comes to integrity, they both differ from each other. While CCF does not have protection against rollback attacks and is therefore vulnerable for them, Nimble specializes on detecting and protecting that kind of attack. Nimble also has the feature of keeping the Trusted Computing Base (TCB) as small as possible. This paper shows that both systems have strengths and weaknesses. Accordingly, both systems have good approaches in different subareas that are all very important for cloud computing, but nevertheless harbor high risks for applications with high requirements of security.

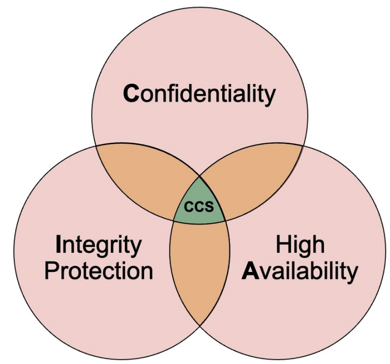

Figure 1: The CIA triad, consisting of the three properties Confidentiality, Integrity Protection, and High Availability, each represented as a circle. A Confidential Cloud Service (CCS) aims to realize all three CIA properties that is the intersection of all circles in the figure.

1 Introduction

Today's applications are increasingly stored in the cloud for more flexibility and scalability. But over time more and more cloud providers get compromised or data is stolen, because providers did not encrypt this data. This is the reason, why finances and health applications that handle sensitive data cannot afford this risk with hosting their applications on the cloud. To make the cloud infrastructure more reliable even for such secure fields, confidential computing has become a new trend.
A Confidential Cloud Service (CCS) is an additional service that can be applied on top of an existing cloud provider [9]. These confidential services are needed when services have high security requirements, which cannot be guaranteed by a normal cloud provider. As the cloud provider or its administration cannot be trusted either, the sensitive data must also be protected from this provider itself.
The CIA triad, shown in Figure 1, describes three important requirements of information security [1, 13]. It contains data Confidentiality, Integrity Protection and High Availability.
Data confidentiality is keeping the data private, which is very important, especially for cloud providers. Challenges are encrypting the data and protecting the encryption keys.
Integrity protection is the insurance of complete and correct code that is not changed by a bad party. Accordingly, it can be considered a prerequisite for confidentiality. But these two characteristics are hard to implement although they are mandatory for cloud computing. This is, because in cloud computing the trusted computing base (TCB) gets bigger and also includes the cloud providers and their infrastructure, so applications in the field of finances, medicine or governmental issues cannot afford to trust this whole TCB.
High availability is required by the fact that people rely on the systems that are on the cloud. Accordingly, they should work, even if failures occur.
Another problem confidential cloud services should handle is every kind of attack. Especially in the cloud, where you cannot even trust the provider, many attacks can happen, above all on big cloud providers. Therefore it is mandatory to detect and protect such attacks like forking attacks, physical attacks or rollback attacks.
This paper introduces two different confidential cloud services, the Confidential Consortium Framework (CCF) and Nimble, and discusses their advantages and disadvantages, especially focusing on rollback attacks.

2 Background

In this section, I will give the necessary definitions and background to understand the following parts.

2.1 Trusted Execution Environments

A Trusted Execution Environment (TEE) is a hardware, e.g. Intel SGX [6], or software-based, e.g. AMD SEV-SNP [12], component where critical code can be run inside a trusted part that is called an enclave. The code and data are confidentiality and integrity protected by malicious parties, hypervisor and the cloud provider. But of course, it is not impossible to change code inside a TEE, so these eventualities should also be kept in mind. Through the concept of remote attestation, it is possible to make sure that the correct code runs inside the TEE [3]. The basic concept is that an attestation key exists with which the TEE can sign its binary and

other components, called the quote. A client can then verify this quote.
As Trusted Execution Environments provide confidentiality and integrity, they are mostly used as a base component of a CCS. Nevertheless, it is important to note that a TEE can also get compromised and therefore violating the CIA properties.

2.2 Distributed Ledger Technology (DLT)

A ledger is a digital register that is made to document transactions or other structural data. The Distributed Ledger Technology (DLT) contains multiple redundant digital ledgers that are decentralized and collect data about transactions in a network and sign them with a cryptographic signature [8]. One use of such a distributed ledger technology (DTB) is a blockchain that is mostly used in the bitcoin technology. However, DLTs are also used for CCS, because they provide confidentiality. Therefore most ledgers are append-only to guarantee the properties of auditability and trustworthiness. This means that data can be written only once in the ledger, but read multiple times. In addition, entries in the ledger cannot be deleted.

2.3 Rollback Attacks

There exist many different attacks against cloud services like forking attacks [4], side channel attacks [5] or rollback attacks [7]. These are all important attacks that should be detected and protected by a confidential cloud service, but in this paper I, will focus on rollback attacks.
In this attack, malicious parties save an older version of the system, restart it, and apply this older version as a new state. So they "roll back" the state of the system. Such an attack can lead to severe consequences. If a bank stores its transactions in a cloud storage, a compromised user can roll back whole transactions. Another example is breaking car keys.
Of course, rollback attacks can also happen on a CCS and therefore it is mandatory to protect this kind of attack. Otherwise whole transactions or configurations can be undone which violates the integrity of the CCS.

3 Confidential Services

In this section, I present two Confidential Cloud Services. The first one is the Confidential Consortium Framework (CCF) [10] and the second one is Nimble [2]. They both provide a structure to guarantee the CIA properties, but have some differences in their design, especially when it comes to rollback attacks. Nimble is specialized in detecting rollback attacks while CCF is a whole framework that can also be used for multiparty applications.

3.1 CCF in a nutshell

The basic idea of CCF is that an untrusted host exists and one or more untrusted users can access the application via different replicated nodes that are responsible for the remaining communication. Providers only need to implement the application logic and make sure to have all necessary endpoints, so that CCF can handle these. Users then only have to access the specific endpoints to use the application. As CCF is especially implemented for multiparty applications, it has a multiparty governance to guarantee confidentiality and integrity. Details are elsewhere, but it consists of a constitution

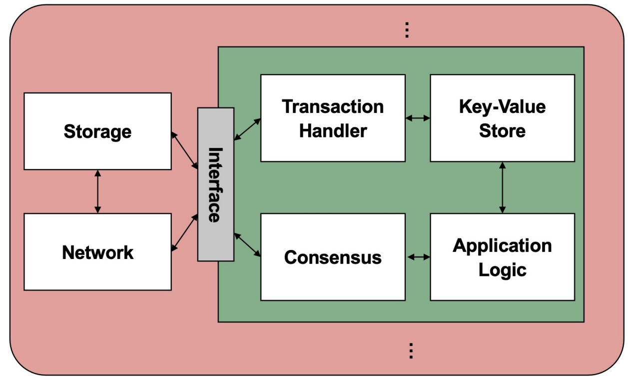

Figure 2: The untrusted host consists of the network, the storage and multiple TEEs (in this figure only one TEE is shown as the inner green box) that can communicate with the outside via an interface. Inside of each TEE that refers to one node in CCF, there is the Transaction Handler, the simple Key-Value Store, the individual Application Logic and the Consensus.

with proposals and ballots that can be individually customized by the application. In Figure 2 the structure of one node is described. The application logic gets the data from the Key-value-store which gets the data from the Consensus which is an explicit layer that is responsible for coordinating CCF in tasks like replicating nodes or making election decisions. The Transaction Handler stores every signature transaction in an append-only ledger that is redundantly stored by one of the nodes that is called primary noce in every other node and the persistent storage. Performing the application logic inside a TEE is fundamental for confidentiality and integrity, and replicating the transactions is necessary for high availability.
While the primary is responsible for handling the users' requests, the remaining nodes are there as a backup. Users can get connected to any node and the specific request is either being forwarded to the primary or handled by the node itself, which is only possible for read-only requests. Every other request has to be forwarded to the primary that executes it.
If a node in use fails, the user can be connected to another node, but if the primary fails there is a primary election that selects a new primary with certain voting criteria. Note that this paper focuses mostly on the functions and challenges of CCF and therefore does not go into detail about how the election works.
User requests are handled as transactions, that have a unique transaction ID. The primary also periodically sends signature transactions that verify all previous transactions as committed. To commit these signature transactions, the primary has to copy it onto at least $\lceil \frac{n-1}{2} \rceil$ nodes, where n is the number of all nodes, to commit it.
Signature transactions can also confirm that there were no malicious changes to the code and thus guarantee the integrity of CCF via merkle proof [11]. Transactions are provided via a merkle tree, where each transaction is represented as a hash value. Every two nodes in the merkle tree are combined with a hash function to a new node. On top of the merkle tree is the merkle root, which is the value that is used for the signature transaction. This process is

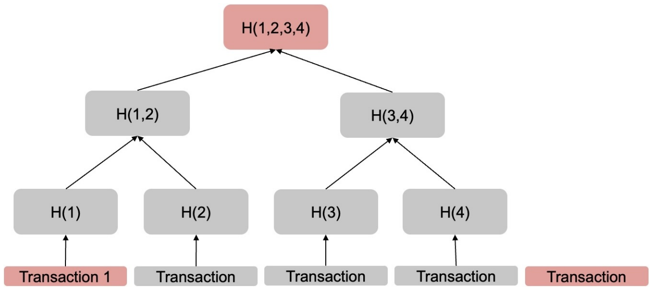

Figure 3: CCF's merkle tree - For every transaction a hash value is computed. Then building a new hash value from the two previous values is repeated until there is only one value left. This is called the merkle root. It is used for the signature transactions that are shown in red signed by the primary.

also shown in Figure 3.

Reconfiguration. Reconfiguration is the procedure when a node fails and is replaced by a new one. This is an important feature to guarantee high availability.
CCF therefore allows to add new or delete old nodes. Reconfiguration is implemented as a transaction. For reconfiguration, a node must request an election and win it. The election is done by a majority quorum. Since a reconfiguration is handled as a transaction, it can also be rolled back. In contrast to Nimble CCF does not detect or protect such an attack on a reconfiguration.
In a disaster scenario, a majority of nodes fail and the system has to handle this.CCF has a disaster recovery protocol, in which the service is started with an older state. Therefore it cannot be guaranteed to be complete. If not all transactions are stored at the ledger before the disaster, it cannot be restored.
Although the integrity of CCF is protected by these signature transactions, the integrity can be violated via rollback attacks. While CCF avoids whole nodes to be rolled back with the help of reconfiguration the ledger is stored outside the TEE and can be attacked this way. This is a huge risk for applications with high safety requirements because whole transactions could be undone. This also leads to certain problems in the reconfiguration that are also discussed later. Therefore I present another confidential cloud service, called Nimble, that has a rollback detection and protection mechanism and also supports reconfiguration to realize better integrity guarantees.

3.2 Nimble in a nutshell?

Nimble is a state machine replication protocol that also tries to ensure the requirement of the CIA triad, but has the focus of preventing rollback attacks. The correctness of Nimble can be proofed, details about this can be found in [2].
Nimble has three important goals, (1) to ensure linearizability, (2) to have a small Trusted Computing Base (TCB) and (3) to guarantee liveness of the cloud service. For the aim to reduce the TCB, they try to use as much of the existing services the cloud provider already uses as possible. The existing cloud storage service that Nimble reuses only has the requirement to be crash fault-tolerant. Therefore they use state machine replication and differ between two main properties that come with this.
Safety means that it must be ensured that every data that can be accessed must be the current data and must not be an older version. This property is enabled via TEEs. In the CIA triad, this would be Confidentiality and Integrity.
Liveness is the High Availability property of the CIA triad. This has not to be ensured via a TEE, because liveness can be easily taken away even in a TEE, and can be handled outside which simultaneously makes the TCB smaller. To guarantee liveness Nimble stores the data redundantly via state machine replication. To guarantee safety the state of each replicated node is appended to a ledger which is stored as a hash chain in an existing storage provider. Inside each TEE there is a trusted state machine that is called endorser. The endorser stores the tail of the hash chain for each ledger. As there are replicated endorsers in Nimble, it can still guarantee liveness, if endorsers fail, as long as there is a majority of working endorsers. To guarantee liveness even if a majority of endorsers fail, Nimble implements reconfiguration.

Reconfiguration. In Nimble it is possible to apply new endorsers. This is also mandatory for the high availability of Nimble. It guarantees both safety and liveness as long as there is a majority of endorsers working. Therefore it is important to add new endorsers if some of them fail because if a majority fails and no new ones are added, Nimble has to give up its liveness. Nimble has also the challenge of implementing the reconfiguration so that the TCB does not get much bigger, which solutions for most systems like in CCF do. There nodes store their identity in each other node with the help of the state replication. This is not possible in Nimble. Therefore each set of working endorsers is stored as a configuration and new endorsers are stored in a new configuration. As a prerequisite for changing the configuration, a majority of endorsers in the previous configuration have called their finalize method and are finished. The identity of a new endorser is created by nimble in the new configuration. The change of the reconfiguration is led by a coordinator in three phases where it (1) finalizes existing. endorsers, (2) initializes new endorsers and (3) activates these new endorsers. While the endorsers can be trusted, the coordinators cannot.

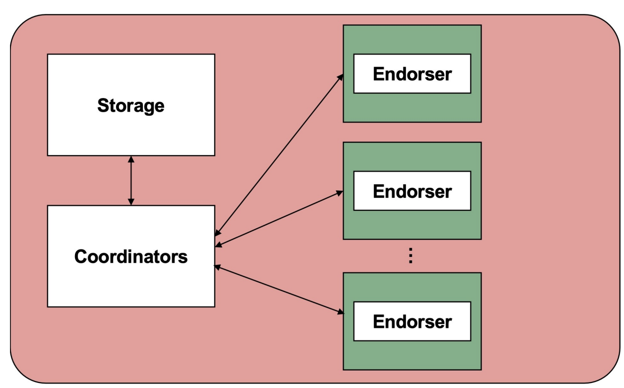

Figure 4: Each node consists of a TEE. Inside the TEE there is the Endorser that saves the tail of the ledger.

4 Rollback Protection

As mentioned before, CCF only provides rollback protection for its nodes. They cannot be rolled back, because a node can never restart in CCF, but has to enter as a completely new node. That means a node can be killed, but never cause a rollback attack on the system. In contrast, the ledger can be rolled back, because it is stored in the persistent storage outside the Trusted Execution Environment and CCF does not provide an additional mechanism to detect rollback attacks. This makes CCF vulnerable and a risk for applications with high requirements of security.
Nimble also stores its state in an existing storage service outside the TEE to use its API, which also makes the TCB smaller. However, in contrast to CCF, it protects this state from compromisation. Nimble, on the other hand, has extended protection against these attacks. To protect rollback attacks they first must be detected. Therefore Nimble presents three categories of rollback attacks:

Stale responses: Stale responses happen when old data is given by the host although there is already newer data. This can easily happen due to the external storage service that cannot be trusted. To prevent stale responses, the state is stored twice. At first, it is encrypted and stored in an external storage servicer, then the state is also stored in the append-only ledger. When an application wants to read the state of the system, it reads both, the one from the storage service, and the one from the ledger, and can detect an attack, when they are not the same. To guarantee liveness a counter is also stored as well in the storage service as on the ledger. The counter on the storage service is always incremented by one and then copied by the ledger. If the system fails before the new state can be appended to the ledger, this can be recognized, as the counter of the ledger is one lower than the one of the storage service. Accordingly, the new state can be added to the ledger, when the system restarts. Nimble uses a linearizable append-only ledger. Linearizability is a criterion that provides strong consistency and is realized via the endorsers.
Remember that an endorser only stores the tail of the ledger, so if data is read from the ledger it can be compared with the endorser. In case it is not the same, a rollback attack did happen or the system failed before the endorser could store the tail of the hash chain. Accordingly, the ledger cannot be rolled back, because the state is provided by the endorser, which does not have a previous state to roll back to.

Synthesized requests: Synthesized requests mean that the provider sends requests that are not sent by the application and stores them. This is handled via signing. The application signs the state and then appends it to the ledger. When the application reads a state from the ledger it then can recognize whether it is a real state with a signature or a synthesized request from a malicious provider.

Replay: When a provider applies older requests to the storage this is called replay. Therefore the position of the stored state in the ledger is also stored with a signature. So the application can check if this signed position matches with the position it is currently stored in the ledger. So Nimble detects the rollback attack and prevents it. Therefore also a replay does not maliciously affect the system.

5 Discussion

In this section I, compare both systems and emphasize their advantages and disadvantages.
Both CCF and Nimble have the aim to guarantee the three CIA properties (1) confidentiality, (2) integrity, and (3) high availability. They also tried to guarantee (1) and (2) with the help of Trusted Execution Environments (TEEs). The problem of both is assuming that TEEs always work correctly as intended. In reality, TEEs can also suffer from different attacks like side-channel attacks or physical attacks and therefore violate the two criteria of the CIA triad. However, TEEs are a relatively new technique and will hopefully be saver in the future.
As mentioned before CCF does not have a rollback detection, because its design ensures that when an attack happens to nodes it cannot affect the system. However, when a rollback attack happens to the distributed ledger, it can affect the system without being noticed. Accordingly, this violates the criteria of integrity, because malicious changes to the ledger can be made, which provides risks for applications running in CCF. As mentioned before, solutions for this integrity problem exist in Nimble. The problem here is that Nimble only focuses on rollback attacks while it completely ignores other kinds of attacks. Therefore also Nimble as an alone standing system would not be safe for applications.
Another challenge is the TCB. When comparing both systems, CCF contains about 55.5 thousand lines of code inside the TEE, while Nimble has only 2.3 thousand lines. With a bigger TCB, the code inside the TEE gets more vulnerable. On the other hand, CCF is much more powerful. While Nimble is specialized in rollback attacks, CCF also provides the possibility to use it for multiparty applications. Despite it would be more efficient if confidential cloud services do not try to put that much code in the TCB to not depend on other services, but reduce the TCB for their field of application.
Unfortunately, this leads to the problem of performance. Nimble's throughput is lower than the C++ implementation of CCF.

6 Conclusion

Comparing both systems shows that both CCF and Nimble have good approaches of making cloud infrastructure more confidential. This is caused by the use of Trusted Execution Environments (TEEs), the distributed ledger technology, the reconfiguration, the multiparty applications on CCF, and the rollback detection and the small TCB of Nimble. However, both systems also have weaknesses. Both are vulnerable to certain attacks, whereby Nimble has, in contrast to CCF, a rollback detection mechanism and rely on the integrity of TEEs that cannot be guaranteed for sure. All in all, there already exist many different Confidential Cloud Services (CCS) with as well some similar as some different approaches. The challenge is, to pick the advantages of the existing services and try to create customized confidential cloud services for different fields of applications. Some might accept a lower performance for the aspect of a smaller TCB, while others do not need that high integrity and therefore choose a better performance.
This work shows how important it is to apply certain CCS to protect applications from the untrustworthy cloud infrastructure and create a trustworthy environment for applications with sensitive data

or processes. Although there already exist many different CCS, they all have strengths and weaknesses and there might be no general solution for all applications running in the cloud. Instead, individual services for certain fields of applications should be developed to focus on their special needs.

References

[1] Michael Aminzade. Confidentiality, integrity and availability–finding a balanced it framework. *Network Security*, 2018(5):9–11, 2018.

[2] Sebastian Angel, Aditya Basu, Weidong Cui, Trent Jaeger, Stella Lau, Srinath Setty, and Sudheesh Singanamalla. Nimble: Rollback protection for confidential cloud services (extended version). Cryptology ePrint Archive, Paper 2023/761, 2023. https://eprint.iacr.org/2023/761.

[3] Alexander Sprogø Banks, Marek Kisiel, and Philip Korsholm. Remote attestation: A literature review. *CoRR*, abs/2105.02466, 2021.

[4] Samira Briongos, Ghassan Karame, Claudio Soriente, and Annika Wilde. No forking way: Detecting cloning attacks on intel sgx applications. In *Proceedings of the 39th Annual Computer Security Applications Conference*, ACSAC '23, New York, NY, USA, 2023. Association for Computing Machinery.

[5] Sebanjila Kevin Bukasa, Ronan Lashermes, Hélène Le Bouder, Jean-Louis Lanet, and Axel Legay. How trustzone could be bypassed: Side-channel attacks on a modern system-on-chip. In *Information Security Theory and Practice: 11th IFIP WG 11.2 International Conference, WISTP 2017, Heraklion, Crete, Greece, September 28–29, 2017, Proceedings 11*, pages 93–109. Springer, 2018.

[6] Victor Costan and Srinivas Devadas. Intel SGX explained. Cryptology ePrint Archive, Paper 2016/086, 2016. https://eprint.iacr.org/2016/086.

[7] Levente Csikor, Hoon Wei Lim, Jun Wen Wong, Soundarya Ramesh, Rohini Poolat Parameswarath, and Mun Choon Chan. Rollback: A new time-agnostic replay attack against the automotive remote keyless entry systems. *ACM Trans. Cyber-Phys. Syst.*, 8(1), jan 2024.

[8] Advait Deshpande, Katherine Stewart, Louise Lepetit, and Salil Gunashekar. Distributed ledger technologies/blockchain: Challenges, opportunities and the prospects for standards. *Overview report The British Standards Institution (BSI)*, 40:40, 2017.

[9] Sascha Fahl, Marian Harbach, Thomas Muders, and Matthew Smith. Confidentiality as a service–usable security for the cloud. In *2012 IEEE 11th International Conference on Trust, Security and Privacy in Computing and Communications*, pages 153–162. IEEE, 2012.

[10] Heidi Howard, Fritz Alder, Edward Ashton, Amaury Chamayou, Sylvan Clebsch, Manuel Costa, Antoine Delignat-Lavaud, Cedric Fournet, Andrew Jeffery, Matthew Kerner, Fotios Kounelis, Markus A. Kuppe, Julien Maffre, Mark Russinovich, and Christoph M. Wintersteiger. Confidential consortium framework: Secure multiparty applications with confidentiality, integrity, and high availability, 2023.

[11] Lum Ramabaja and Arber Avdullahu. Compact merkle multiproofs, 2020.

[12] AMD Sev-Snp. Strengthening vm isolation with integrity protection and more. *White Paper, January*, 53:1450–1465, 2020.

[13] Michael E Whitman, Herbert J Mattord, et al. *Principles of information security*. Thomson Course Technology Boston, MA, 2009.