Cholula: Fast, Fault-tolerant, and Strongly Consistent Off-chain Object Storage

David Shen[1], Cole Dumas[1], Callie Sardina[1], Lewis Tseng[1], Moayad Aloqaily[2]

[1]Boston College, MA, USA
[2]Mohamed Bin Zayed University of Artificial Intelligence (MBZUAI), UAE
E-mails: {shendc, dumasca, sardinac, lewis.tseng}@bc.edu, maloqaily@ieee.org

***Abstract*—Emerging security technologies such as Blockchain are prominent to secure data. Blockchain inherits the advanced principles of cryptography and decentralization. This paper explores an appropriate design for distributed object storage systems for off-chain data. Our system Cholula has the following salient features: (i) Strong consistency; (ii) Optimal fast-path latency (1 RTT when there is no conflicting write); (iii) Tolerance of Byzantine servers; (iv) Security (preventing from an imposter attack); and (v) Censorship-resistance. We present our design and evaluation in this work. We demonstrate that Cholula has a better performance than the state-of-the-art object storage system Giza (the system behind Microsoft OneDrive) and Cassandra in the context of geo-replicated off-chain data.**

***Index Terms*—Object storage, Fault-tolerance, Off-chain storage, Byzantine attack, Geo-replication**

I. Introduction

Blockchain provides a secure, decentralized system for transactions between potentially untrustworthy nodes. Blockchain records data in an transparent, append-only, timestamped manner which is distributed to all the participating nodes within the network [1]. Blockchains are becoming increasingly prevalent across many industries. More interestingly, intelligent Blockchain systems are promising for the use of sophisticated communication and services for bridging the traditional gap of using blockchain applications [2]. Useful applications of blockchain include: increasing financial transaction trustworthiness, providing traceability within the supply chain, supporting the creation of a global currency, and ensuring information security within the healthcare industry – to name a few.

Blockchain is known to have limited scalability. In order to assure integrity and security, data (or a block of transactions) is replicated at every node and the order of blocks (and contents in the blocks) need to be verified and agreed upon by each node, however, this can hinder the scalability. Due to the limitation, it is too expensive for current smart contracts to generate large quantities of data on-chain. However, there is a need to exchange a large amount of data.[1] Off-chain storage provides a solution for scaling transactions. Data can be stored in off-chain nodes (or off-chain databases) in order to unburden the blockchain and increase efficiency, while retaining the traceability and trustworthiness of blockchain [4], [5]. Thus, off-chain storage provides a more efficient means of storing ledger states and executing smart contracts used in general-purpose transactions.

Current off-chain storage solutions include Swarm, Hypercore, SAFE, Storj, and Arweave [4], [5]. However, limitations concerning storage, mutability, privacy, access control, and use case applications persist [4], [5]. We defer a detailed discussion on the limitations to Section II-B and other related work to Section V.

In this paper, we present our system, Cholula – a fast, fault-tolerant and strongly consistent off-chain object storage. Our system additionally addresses three limitations of off-chain storage: imposter attack, immutability, and censorship, which will be subsequently discussed.

In order to employ Blockchain solutions effectively in these applications, strong consistency is necessary. Strong consistency refers to the property that all nodes must observe the same total order of operations – read and write. To accomplish strong consistency and preserve the total order, nodes need to agree on the ordering. Strong consistency [6] helps ensure correctness. Even though it also affects the performance and availability properties e.g., the CAP theorem [7], Cholula optimizes the fast-path performance to achieve 1 RTT (round-trip) delay, which is optimal in such storage systems. Fast path occurs when operations commit in a single communication step.

We also experimentally demonstrate that Cholula outperforms Giza [8], the state-of-the-art object storage system (which supports Microsoft OneDrive), and Cassandra [9], a popular NoSQL in the context of geo-replicaiton when there is no conflicting write (the common case for our targeting scenarios).

II. Preliminaries and Key Observations

A. Model

The off-chain storage system consists of n replicas, which are static, and n_c clients such that clients and replicas are communicate in an asynchronous message-passing networks. We assume $\leq f$ replicas may become Byzantine faulty. That is, Byzantine replicas can behave arbitrarily, including sending malicious messages and tampering stored content. The clients are assumed to be crash-prone. Links are assumed to be asynchronous (messages could be delivered with an arbitrary delay) and reliable. A reliable link has two properties: (i)

[1]Even with a recent crash in the crypto market, storing 1GB of data on-chain still costs around USD 30.48 Million on Ethereum at the time this paper was written [3].

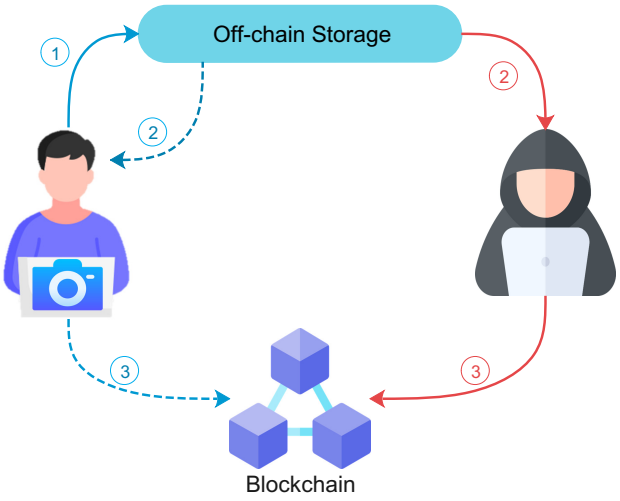

Fig. 1: Illustration of an imposter attack

eventually, messages would be delivered from a fault-free sender to a fault-free receiver, and (ii) a fault-free receiver receives a message if and only if a sender has sent the message.

B. Limitations of Current Off-chain Storage Systems

In the following discussion, we will use NFT (non-fungible token) minting as a working motivating example, as it is arguably one of the most popular Blockchain applications at present. In general, NFTs are minted in three steps [10]: (1) storing the image to an off-chain storage, e.g., IPFS; (2) obtaining the content hash; (3) minting an NFT with that hash embedded in the NFT metadata. We discuss three issues which may occur with this model.

There are three limitations of using current off-chain storage systems:

- *Imposter attack*: If the second step or third occurs slowly at the original artist, there is the risk of an attacker stealing and minting an NFT before the true artist can do so. The attacker would then own the NFT. Figure 1 presents the illustration of this attack. Dashed lines represent slow links whereas red lines represent the malicious imposter that steal the artworks stored on the off-chain storage. We defer discussion on the relevance and practicality of the attack and NFT minting to Appendix A.
- *Immutability*: A current limitation to a decentralized off-chain storage, e.g., IPFS (The Interplanetary File System), is the requirement of immutability. Several artists cannot collaborate on or modify the same artwork which has been stored in the off-chain storage because of the immutability property.
- *Censorship*: The issue of censorship also arises if a centralized off-chain storage system removes the content.

Our system addresses the three aforementioned downfalls of off-chain storages. To protect against an imposter attack, we use erasure coding, making it necessary for an attacker to obtain a subset of these fragments in order to steal the data. As long as the attacker cannot compromise that many replicas, the image/artwork remains safe. Typical decentralized off-chain storages do not use coding for performance issues.

We address immutability by adopting the versioned object storage – every write/mutation creates a new version. The off-chain storage employs a key-value structure and creates a versioned object for each object creation or mutation. This protects against purposeful or accidental manipulation of the data.

Our system employs a cloud-of-clouds [11] model as opposed to a decentralized peer-to-peer model (akin to IPFS) to evade censorship. Our model mimics a decentralized structure through the use of several different cloud service providers. Our approach makes censorship harder because various clouds may or may not censor varying types of data, especially for cloud providers located in different countries. Because of erasure coding, each cloud does not store a full copy of the data. Therefore, even if a shard of data is removed from storage in one cloud, the source data can still be retrieved through shards on other clouds.

C. Target Workload

In our target scenarios, there are two phases:

- Phase 1: before the NFT is minted, there are a small number of artists (clients to our off-chain storage) that may read and write concurrently. That is, n_c is small.
- Phase 2: after the NFT is minted, there are a large number of readers. There will be no writer, since the artwork and the NFT is already completed and should remain immutable. Note that in the typical NFT applications, others still read the artwork from the off-chain storage.

Moreover, replicas or storage servers could be compromised or have a non-zero incentive to censor the hosted content, e.g., pressures or requests from the government. Hence, we model them to be susceptible to Byzantine faults. That being said, compared to the peer-to-peer systems, cloud providers usually provide more trustworthy security measures and access control. This also helps Cholula prevents from common security attacks, e.g., denial-of-service or man-in-the-middle attacks.

In the case of clients, as typical in most Byzantine storage systems [11]–[13], we assume the clients are crash-prone. Particularly, the artists are assumed to be fault-free, since they have a strong incentive to be reboot and recover to complete the task and artwork.

D. Properties of Off-Chain Storage Systems

Cholula satisfies the following properties:

- Strong consistency: our system satisfies linearizability [6]. Roughly speaking, all the operations observed by the clients appear to occur instantaneously and follow the same total ordering that respect the real-time ordering.
- Optimal fast-path latency: when there is no conflicting write operation, both read and write operations can be completed in 1 RTT.
- Fault-tolerance: it tolerates up to f Byzantine servers.

- Security: it prevents from the imposter attack outlined above, as long as up to f servers are corrupted, and other security attacks ensured by common cloud providers.
- Censorship-resistance: any single cloud provider cannot directly remove the stored content.

III. Cholula: Design

Our design is inspired by Giza [8]. We also develop our system and Giza on top of Cassandra [9]. Since Giza is Microsoft's proprietary system, we have to clone one based on the description in the paper. Thus, we describe the concept of Giza with our integration of Cassandra first, followed by the key technical changes to derive Cholula. For brevity, we focus on the metadata replication and the write operation, since using erasure coding for the data part is quite straightforward, and read operation is symmetric to a write. Roughly speaking, metadata contains the necessary control information associated with the data (or coded shard) itself.

A. Fast and Slow Write Paths

1) Fast Write - FastPaxos: Fast Paxos [14] is used for the fast write path, which Giza attempts first and resolves in one round trip if there is no contention. First it reads the *highest_known_committed_version* field to retrieve the latest version received locally, and increments it to use as the new version.

It then sends a PreAccept request for its current metadata value, which involves a LWT (lightweight transaction) that ensures there is not already a value PreAccepted for the object and that the proposed version is greater than any committed version before writing the node's ID into *preaccepted* and the metadata value into *preaccepted_value*.

If the PreAccept succeeds for a fast quorum (greater than $3N/4$), then the result is returned to the client and the value is committed asynchronously, where it sets the committed value of that version, adds it to the *known_committed_versions* set, and updates *highest_known_committed_version*. Finally, it cleans up the PreAccept request by clearing the *preaccepted* and *preaccepted_value* field using another LWT, to verify that the PreAccepted field still contains the current node's ID[2]. This is executed asynchronously and will take an additional round trip, but the time needed is unimportant because the Giza can already guarantee that the value is saved, and responds to the client before executing those tasks.

If the PreAccept fails, either because another value has already been PreAccepted (this is the case of contention), or the version proposed has already been committed, Giza moves to the slow write path.

2) Slow Write - Classic Paxos: The slow write path involves the class Paxos algorithm with the standard Prepare/Promise, Propose/Accept phases. However, this is done using Cassandra and LWT. These are executed on the highest version of the object that is not committed.

The first step is the Prepare request. Giza queries its local database for the version's *highest_ballot_seen*, increments it, and executes a LWT that writes the incremented ballot into *highest_ballot_seen* if there has not been a higher ballot. If the Prepare request fails to receive a quorum of positive responses, it restarts with the fast write path.

At the same time, Giza performs a query on the other nodes, to read their respective *preaccepted*, *preaccepted_value*, *highest_ballot_accepted*, and *highest_value_accepted* fields. Once it has received a quorum of responses, it picks a value to propose based on three cases:

Case 1: If there were any highest accepted ballots in the responses, Giza picks the value of the largest one to propose.

Case 2: Otherwise, if there were any PreAccepted values in the responses, Giza picks the one seen most frequently to propose.

Case 3: Finally, if both of the above fail, there is no real contention. Giza proposes its own value.

Once the value is picked, Giza performs the Propose/Accept phase with another LWT: if the current ballot is greater than the highest seen and highest accepted, and the version has not yet been committed, it sets *highest_ballot_accepted* to the current ballot and *highest_value_accepted* to the picked version.

If it fails to receive a quorum of successful writes, or if the value proposed is not its own, Giza restarts with the Fast Write path, using the original value provided by the client. Otherwise, the write is complete.

B. Database Design

The structure of the database is crucial to how Giza works. The database is created with the schema as outlined in Figure 2 and Figure 3, containing both static and dynamic fields.

```
TABLE state (
    object_id uuid,
    version int,
    value blob,
    committed boolean,
    highest_ballot_seen bigint,
    highest_ballot_accepted bigint,
    highest_value_accepted blob,
    preaccepted int STATIC,
    preaccepted_value blob STATIC,
    known_committed_versions
                set<int> STATIC,
    highest_known_committed_version
                int STATIC,
    PRIMARY KEY (object_id, version)
);
```

Fig. 2: Database schema of Giza and Cholula

- *object_id* and *version* together create the primary key for each version of each object in Giza. They combine

[2]The design in the Giza paper did not explain that PreAccept used a field in the database (for storing metadata). Considering how one of the major unique points of Giza was its use of the database for Paxos states, we thought it would logically follow that it would be used for the Fast Paxos states as well.

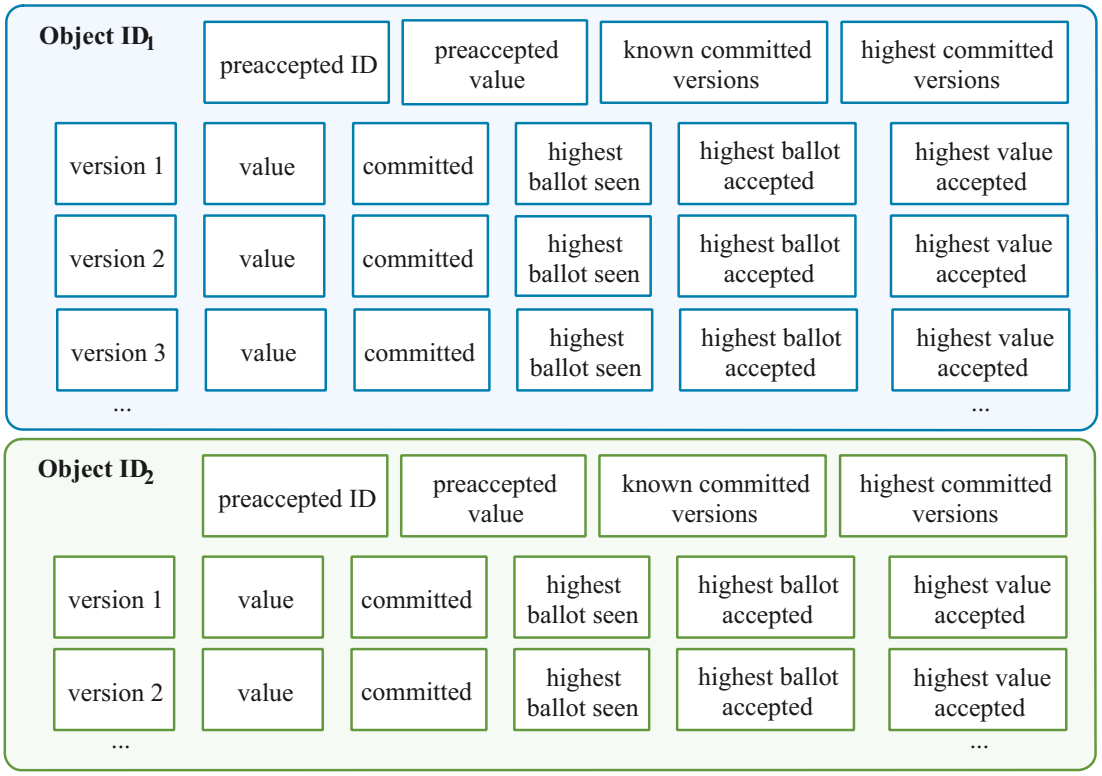

Fig. 3: Visualization of the database design. In this example, we have two objects, each with several versions. The first row of each object is used for control information, whereas the later rows are data regarding each version (of the object).

to create a compound primary keywith *version* as the selector within *object_id*.

- The static fields *preaccepted* and *preaccepted_value* are used for the Fast Write path, to store the value that is PreAccepted and the ID of the node that sent the request.
- The static field *known_committed_versions* stores the set of versions that have been successfully committed, and *highest_known_committed_version* stores the highest committed version, for simpler lookup when selecting a new version number.
- *highest_ballot_seen*, *highest_ballot_accepted*, and *highest_value_accepted* store the Paxos states in the Slow Write path for each version of each object.
- When a value is finally committed, *committed* is set to True and *value* receives the committed value.

C. Cholula

Obviously, Giza does not satisfy our use scenarios because it does not tolerate Byzantine servers. To provide efficient Byzantine fault-tolerance, we push the work to the clients to resolve conflicts (i.e., handling the slow path). As is outlined in Section II-C, the number of writer clients is expected to be small, making this design a reasonable tradeoff.

The high-level architecture of Cholula is presented in Figure 4. The top boxes are Cholula replicas/servers located at different data centers (DC), which has three roles: (i) Cholula proxy that implements the replication logic; (ii) metadata storage that stores the control information; and (iii) object storage that stores the coded element of an object. Bottom circles are clients which communicate with both replicas and clients. Now, we describe the key differences between Cholula and Giza.

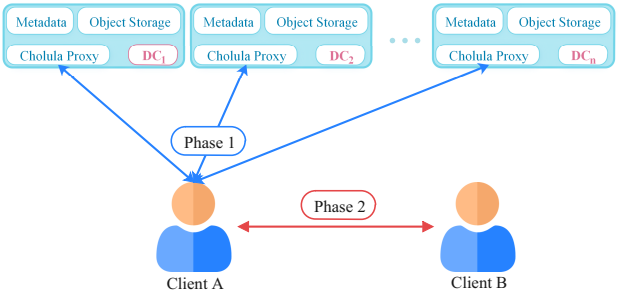

Fig. 4: Cholula: System Architecture. In this illustration, there are two clients and n data centers DC_1 to DC_n. Each data center has Cholula proxy that interacts with metadata replication and object storage. In our design, only phase 1 (fast path) is between clients and data centers, whereas phase 2 is between clients.

- Fast path (phase 1): clients contact a majority of nodes (instead of $3N/4 + 1$ as in Giza). If all the responses match, then clients use the same approach to commit.
- Slow path (phase 2): if a response does not match, then

	CA	VA	IR	OR	JP
CA	0.2				
VA	72	0.2			
IR	151	88	0.2		
OR	59	93	145	0.2	
JP	113	162	220	121	0.2

TABLE I: RTT (in ms) between VMs in emulated geographic regions [15]. CA stands for California, VA for Virginia, IR for Ireland, OR for Oregon, and JP for Japan. The RTT represents typical ping latency between different Amazon Web Service regions.

clients contact each other and use voting to agree on a version for each conflicting write. Note that in this case, it is possible that a Byzantine server lies about the existence of a conflicting writes. However, by assumption, all the writer clients are fault-free, so such a malicious behavior can always be detected.

IV. Evaluation

We present our evaluation of Cholula and two state-of-the-art storage systems Microsoft Giza [8] (the core of Microsoft OneDrive) and Cassandra [9] (one of the most popular NoSQL for big data workload).

A. Experiment Setup

To analyze the performance of Giza and Cholula, we implement Giza and the fast path of Cholula on top of Cassandra. Since the data path of storing coded element to a local object storage is similar, we focus on the metadata replication in our evaluation. We use Cassandra to realize this part.[3] The layers of Giza and Cholula on top of Cassandra are implemented in Go and all three systems using Cassandra internal engine as a core, which make the comparison fair between systems.

We ran three separate experiments, one for each system. With the exception of the protocol used, each experiment had an identical setup consisting of five nodes representing five different regions: California (CA), Virginia (VA), Ireland (IR), Oregon (OR), Japan (JP). Each node pair has a customized artificial latency, chosen to emulate the latency between actual ones between Amazon Web Service sites [15]. Table I shows the RTT values for each pair of nodes.

The machines were setup on CloudLab using 5 m510 nodes, which had 8 core Intel Xeon D-1548 processors. The CloudLab profile on the GitHub was used to create the nodes on a LAN. Artificial delay between was added using Linux's Traffic Control (tc) to add delays to packets on all nodes with filters on different IPs. They ran a single instance of the given protocol, which was set up using the script provided on the GitHub. Each metadata object was 11 bytes, and the experiments ran for 3 minutes.

For each experiment, we measure the performance with no contention so that all the systems have the best performance (i.e., on the fast path). As analyzed in Section II-C, this is a common case for our target workload. We deploy one client at the CA location to write data (randomly selected from one of the 128 objects) sequentially. The read and write are symmetric, so we only report the latency distribution of write operations, as shown in Figure 5.

[3]Our code can be found at https://github.com/pantherman594/Giza/

B. Summary of Our Findings

In the non-contention case (when there is no conflicting writes), both Giza and Cholula take the fast path. This is why we observe the straight line of the latency cumulative distribution function (CDF) in the figure. Due to different fast quorum size, Cholula has latency around 96ms and Giza around 125ms. This is because the Cholula client at CA only needs to wait for the response from the DC located at VA, whereas Giza client at CA waits for the DC located at JP. Note that the latency is slightly larger than the ping latency reported in Table I because of the latency in accessing Cassandra table (at local data center), as outlined in Section III.

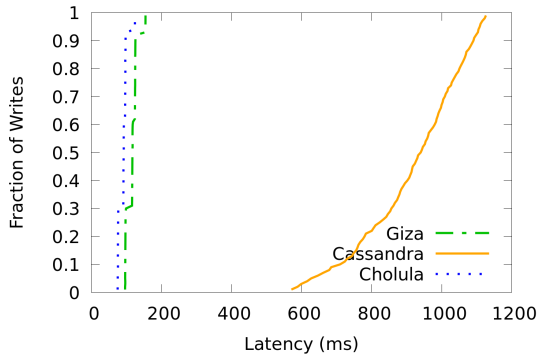

Fig. 5: Latency CDF with non-contention cases (fast path performance). Both Cholula and Giza complete in 1 RTT; however, Cholula is faster because of a smaller fast quorum.

V. Related Work

Eisenring evaluates various off-chain storage systems for use in Blockchain Signaling System and posits IPFS as the most suitable solution when compared with Swarm, Storj, and CoAP [4]. No security or privacy is considered in his work. Daniel and Tschorsch [5] compare IPFS and closely related peer-to-peer systems off-chain storage solutions:

- *Swarm*: A P2P distributed content storage and retrieval system with an incentive mechanism based upon the Ethereum ecosystems. Drawbacks to Swarm include potential storage issues, available content limitations, and its dependency on Ethereum.
- *Hypercore*: A P2P system consisting of append-only Hypercore logs, used for sharing large, mutable data. Limitations of Hypercore include privacy concerns and lack of detailed access controls.
- *Storj*: A decentralized cloud storage framework with encrypted shards, erasure coding, and a reputation system to protect against Byzantine attacks. Concerns regarding

Storj include increased centralization and privacy leaks. That is, the replica selection is out of the control of clients.

- *Arweave*: An on-chain data storage system for permanent data. Arweave offers increased scalability solutions, but retains storage limitations as an on-chain storage system, and lacks mutability.

None of the system supports the features Cholula provides.

Tschirner et al. [16] confronts the challenge of data privacy by balancing the tradeoffs between data control and transparency with the introduction of moving smart contracts, which provide a means of keeping data localized while trusted nodes retain the ability to access the data and execute smart contracts. We do not focus on the computation aspect. Plus, to provide availability and scalability, Cholula replicates data to multiple nodes.

SlimChain addresses the poor scalability of Blockchain [17]. SlimChain is a stateless system which stores ledger states and executes smart contracts in off-chain nodes to relieve the blockchain of storage requirements and increase efficiency. SlimChain requires an integration with a customized on-chain transaction validation and state commitment, whereas Cholula is compatible with any existing smart contracts that use off-chain storages.

There are also works on Byzantine storages, e.g., [11]–[13]. However, these systems do not support strong consistency, nor are they designed with off-chain usage in mind. Typically, they do not prevent from imposter attack, nor support censorship resistance.

VI. Conclusion

This paper presents Cholula, an off-chain versioned object storage system with several desirable properties. We experimentally demonstrate that Cholula provides better performance than state-of-the-art systems (designed for non-Byzantine use cases). This is mainly because the design of Cholula is optimized for the common use scenarios for off-chain storages.

Acknowledgements

Authors from Boston College were partially supported by National Science Foundation award CNS-1816487. Any opinions, findings, and conclusions or recommendations expressed here are those of the authors and do not necessarily reflect the views of the funding agencies or the U.S. government. The authors would also like to acknowledge useful discussion on NFTs with Jiahong Li.

Appendix A
Remark on NFT Minting and Imposter Attack

To keep the discussion concise, we omit some steps in most NFT use scenarios, including (i) uploading metadata along with the content hash (or CID), which includes image attributes, to the off-chain storage; (ii) intentionally keeping an image as a place holder until the reveal phase; and (iii) changing the baseURI so that it links to the correct set of images during the reveal phase.

Since Web3 and NFTs are mostly community-driven, one could argue that the imposter attack will not affect popular and well-known artists. This is because imposter could easily be debunked and cannot make any monetary gains. However, such an imposter attack is such a low cost one, especially for images upload to IPFS or similar P2P storages where host machines could easily be compromised. With this vulnerability, new or uprising artists could feel thwarted and mature ones would feel discouraged if they need to constantly convince the communities on the correct set of images.

References

[1] Avi Silberschatz, Henry F. Korth, and S. Sudarshan. *Database System Concepts, Seventh Edition*. McGraw-Hill Book Company, 2020.

[2] Ismaeel Al Ridhawi, Moayad Aloqaily, and Fakhri Karray. Intelligent blockchain-enabled communication and services: Solutions for moving internet of things devices. *IEEE Robotics Automation Magazine*, 29(2):10–20, 2022.

[3] Steffen Lehmann. Minimizing data storage cost on the ethereum network. https://proderivatives.com/blog/2019/5/10/minimizing-data-storage-cost-on-the-ethereum-network, 2019.

[4] Lukas Eisenring. Performance analysis of blockchain off-chain data storage tools, 1 2018. Bachelor's Thesis.

[5] Erik Daniel and Florian Tschorsch. IPFS and friends: A qualitative comparison of next generation peer-to-peer data networks. *IEEE Commun. Surv. Tutorials*, 24(1):31–52, 2022.

[6] Maurice P. Herlihy and Jeannette M. Wing. Linearizability: A correctness condition for concurrent objects. *ACM Trans. Program. Lang. Syst.*, 12(3):463–492, 1990.

[7] Eric Brewer. A certain freedom: Thoughts on the CAP theorem. In *Proc. ACM SIGACT-SIGOPS Symposium on Principles of Distributed Computing (PODC)*, pages 335–335, 2010.

[8] Yu Lin Chen, Shuai Mu, Jinyang Li, Cheng Huang, Jin Li, Aaron Ogus, and Douglas Phillips. Giza: Erasure coding objects across global data centers. In Dilma Da Silva and Bryan Ford, editors, *2017 USENIX Annual Technical Conference, USENIX ATC 2017, Santa Clara, CA, USA, July 12-14, 2017*, pages 539–551. USENIX Association, 2017.

[9] A. Lakshman and P. Malik. Cassandra: a decentralized structured storage system. *ACM SIGOPS Operating Systems Review*, 44(2):35–40, 2010.

[10] IPFS Docs. Mint an nft with ipfs. https://docs.ipfs.io/how-to/mint-nfts-with-ipfs/, 2021.

[11] Marko Vukolic. The byzantine empire in the intercloud. *SIGACT News*, 41(3):105–111, 2010.

[12] Kishori M. Konwar, Saptaparni Kumar, and Lewis Tseng. Semi-fast byzantine-tolerant shared register without reliable broadcast. In *Proceedings of the 40th IEEE International Conference on Distributed Computing Systems ICDCS 2020*. IEEE, 2020.

[13] Alysson Neves Bessani, Miguel Correia, Bruno Quaresma, Fernando André, and Paulo Sousa. Depsky: Dependable and secure storage in a cloud-of-clouds. *ACM Trans. Storage*, 9(4):12:1–12:33, 2013.

[14] Leslie Lamport. Fast paxos. *Distributed Comput.*, 19(2):79–103, 2006.

[15] Matthew Burke, Audrey Cheng, and Wyatt Lloyd. Gryff: Unifying consensus and shared registers. In Ranjita Bhagwan and George Porter, editors, *17th USENIX Symposium on Networked Systems Design and Implementation, NSDI 2020, Santa Clara, CA, USA, February 25-27, 2020*, pages 591–617. USENIX Association, 2020.

[16] Simon Tschirner, Shashank Shekher Tripathi, Mathias Roeper, Markus M. Becker, and Volker Skwarek. Moving smart contracts - A privacy preserving method for off-chain data trust. *CoRR*, abs/2205.08440, 2022.

[17] Cheng Xu, Ce Zhang, Jianliang Xu, and Jian Pei. Slimchain: Scaling blockchain transactions through off-chain storage and parallel processing. *Proc. VLDB Endow.*, 14(11):2314–2326, 2021.