ENGINEERING IN ADVANCED RESEARCH SCIENCE AND TECHNOLOGY

ISSN 2350-0174
Vol.02, Issue.11
November -2020
Pages: -231-240

THE FIRST CERTIFICATE LESS PUBLIC VERIFICATION SCHEME AGAINST PROCRASTINATING AUDITORS BY USING BLOCK CHAIN TECHNOLOGY

[1] RAYANAPETA MOUNIKA [2] KAMBHAM SALIVAHANA REDDY, M. tech (Assistant professor)
[1,2]Global College of Engineering and Technology, Dept. of CSE

ABSTRACT– Blockchain is challenging the status quo of the central trust infrastructure currently prevalent in the Internet towards a design principle that is underscored by decentralization and transparency. In ideal terms, blockchain advocates a decentralized, transparent, and more democratic version of the Internet. Essentially being a trusted and decentralized database, blockchain finds its applications in fields as varied as the energy sector, forestry, fisheries, mining, material recycling, air pollution monitoring, supply chain management, and their associated operations. The preparation of cloud storage services has important edges in managing knowledge for users. However, it conjointly causes several security issues, and one amongst them is knowledge integrity. Public verification techniques will change a user to use a third-party auditor to verify the info integrity on behalf of her/him, whereas existing public verification schemes square measure prone to procrastinating auditors WHO might not perform verifications on time. moreover, most of public verification schemes square measure created on the general public key infrastructure (PKI), and thereby suffer from certificate management downside. during this paper, we tend to propose the primary certificate less public verification theme against procrastinating auditors by victimization blockchain technology. The key plan is to need auditors to record every verification result into a blockchain as a dealing. Since transactions on the blockchain square measure time-sensitive, the verification will be time-stamped when the corresponding dealing is recorded into the blockchain, that permits users to see whether or not auditors perform the verifications at the prescribed time

KEYWORDS: Blockchain, Cloud, public Key, data Integrity

1 INTRODUCTION With cloud storage services, users supply their data to cloud servers and access that data remotely over World Wide Web. These services supply users Associate in nursing economical and versatile because of manage their data, whereas users ar free from serious native storage costs. Although users relish nice blessings from these services, data outsourcing has together incurred necessary security issues. One of the foremost necessary security problems is data integrity. In distinction to ancient data management paradigm, where users store their data domestically, users would not physically own their data once having outsourced the data to cloud servers. Therefore, users ar

unceasingly distressed regarding the data integrity, i.e., whether or not or not the outsourced data is well maintained on cloud servers. The integrity of outsourced data is being place in peril in apply. As an example, the cloud servers would possibly unceasingly conceal incidents of data corruption for good name, or would possibly delete a section of data that is never accessed to reduce the storage costs. Moreover, Associate in Nursing external resister would possibly tamper with the outsourced data for cash or political reasons. Therefore, the integrity of outsourced data need to be verified periodically. The verification is also performed by the users themselves. Public verification techniques amendment users to supply the data integrity verification to a fervent third-party auditor. The auditor periodically checks the data integrity, and informs the users that the data might even be corrupted once the checking fails. In most of public verification schemes, the auditor is assumed to be honest and reliable. If the auditor is compromised, these schemes would be invalid. as an example, Associate in Nursing unaccountable auditor would possibly unceasingly generate AN honest integrity report whereas not taking part in the verification to avoid the verification costs. In such the manner, the auditor is sort of non-existent. moreover, a malicious auditor would possibly conspire with the cloud servers to urge a bias verification result to deceive the users for profits. to substantiate the protection inside the case that the auditor is compromised, the users ar required to audit the auditor's behaviors once each verification and additionally the auditor records the information accustomed verify the data integrity, that allows the user to audit the validity of the auditor's behavior.

2. RELATED WORK:

1. Blockchain

A blockchain is essentially a fault-tolerant distributed database of records of a public ledger of all transactions that have been executed and shared among participating parties[10]. Each transaction in the public ledger is verified by a consensus of a majority of participants in the system. Blockchain provides a tool for increasing data integrity using a combination of cryptography and consensus[11]. As such, there is no central server point of control. The in formation is stored in blocks, Each block contains transactions, a timestamp and a link to the previous block, and is cryptographically protected. What makes it secure is:

• Public key infrastructure: It uses public/private key encryption and data hashing to store and exchange data safely.

• Distributed ledgers: There is no central authority to hold and store the data, it removes the single point of failure

• Peer to peer Network: The communication is based on the P2P network architecture and inherits the decentralized characteristics.

• Cryptography: Blockchain uses a variety of cryptographic techniques, hash functions, Merkle trees, public and private key[12]. It is difficult to alter the blockchain, to make a modification, it is necessary to succeed in a simultaneous attack on more than 51% of the participants.

• Consensus Algorithm: The rules which the nodes in the network follow to verify the distributed ledger. Consensus algorithms are designed to achieve reliability in a system involving multiple unreliable nodes. A consensus of the nodes validates the transactions. Choice of consensus algorithm has significant implications for performance, scalability, latency and other Blockchain parameters. The consensus algorithms must be faulttolerant[13].

• Transparency: Each transaction is visible to anyone with access to the system. If an entry can not be verified, it is automatically rejected. The data is therefore wholly transparent. Each node of a blockchain has a unique address that identifies it. A user may choose to provide proof of identity to others.

In blockchain, the shared data is the history of every transaction ever made. The ledger is stored in multiple copies on a network of computers, called nodes. Each time someone submits a transaction to the ledger, the nodes check to make sure the transaction is valid. The transaction is being created in the chained data structure of blockchain, a new timestamp will be recorded at the same time, and any modification of data created before will not be allowed anymore. A block is a data structure which consists of a header and a list of transactions. Each transaction is generally formed as trx:= (M, Signature) where M:= (PKSender, receiver, data); PKSender denotes the public key of a sender (address of the sender is derived from the PKSender), and receiver is the address of a receiver, Signature:=Sign SKSender (H(M)) where SKSender is the private key of the sender and H() is a secure hash function. A subset of them competes to package valid transactions into blocks and add them to a chain.

The owners of these nodes are called miners and the miners who successfully add new blocks to the chain earn a reward. A cryptographic hash unique to each block and a consensus protocol makes it tamper proof[13].

To ensure the integrity of data stored on an untrusted server, Juels et al. [30] proposed the "proofs of retrievability" (POR) technique. However, in [30], public verification is not considered, and hence the data owner needs to verify the data integrity periodically. This requires the data owner to keep online for verification. As such, the data owner has to bear heavy communication and verification burden to retrieve and use the data. At the same time, Ateniese et al. [16] proposed the "provable data possession" (PDP) technique, which is the first scheme that considers the public verification, where the data owner is able to employ a third-party auditor to check the data integrity on behalf of her/him. Later, Shacham et al. [14] proposed the first compact POR scheme with full proofs of security against arbitrary adversaries in the strongest model proposed by Juels et al. [30]. Following the Shacham et al.'s work, several public verification schemes have been proposed [9], [13], [15], [32], [46]. These schemes are built upon a homomorphic signature technique.

Privacy preserving public verification has also attracted attentions in the recent literature. A privacy-preserving public verification scheme enables the auditor to verify the integrity of oursourced data without learning the content of the data. Most of existing privacy-preserving schemes, such as [9], [15], [31], [32], require the cloud server to utilize a random mask to blind the proof information such that the

auditor can check the validity of the proof information without extracting the data content. For our scheme, to protect users' data against the auditor, we encrypt the data before generating the tags. Since in CPVPA, we consider that the auditor may collude with the cloud server, the cloud server would straightforward send the data to the auditor to violate the data privacy of users.

The above schemes are built on certificate-based cryptography and thereby are confronted with the certificate management problem. To avoid managing certificates in a public verification scheme, some schemes [10], [47] constructed on the identity-based signature schemes were proposed. However, these schemes are subject to an inherent drawback: the key escrow problem [28]. Wang et al. [23] proposed the first certificateless public verification scheme. Then some certificateless public verification schemes with enhanced security were proposed [18], [24]. These schemes are constructed on the homomorphic certificateless signature schemes. Therefore, the auditor in these schemes does not need to manage the users' certificates without confronting with the key escrow problem. Furthermore, for the existing schemes, the auditor is assumed to be honest and reliable. However, this is a very strong assumption as corruption of auditors could happen in practice. Recently, Armknecht et al. [17] proposed the first PoR scheme that thwarts malicious auditors, and Zhang et al. [18] proposed the first public verification scheme with resistance against malicious auditors. These schemes cannot resist procrastinating auditors who may not perform the data integrity verification on schedule. A procrastinating auditor can deviate from the primary objective of public verification that detect the data corruption as soon as possible. It is worth clarifying that resistance against procrastinating auditors is vitally important for public verification schemes in practice. More detailed survey on public verification of data integrity can be found in [48].

In most of existing public verification schemes [14], [16], [30], auditors are assumed to be honest and reliable. This means that the auditor would honestly follow the prescribed schemes, and performs the verification reliably2 . These schemes cannot resist malicious auditors. The most trivial attack a malicious auditor can perform is that it always generates a good integrity report without verifying the data integrity to avoid the verification burden. To thwart such attacks, the user is able to audit the auditor's behavior at the end of each epoch. However, a more tricky attack still exists in the mechanism: the auditor colludes with the cloud server, and always generates bias challenging messages such that only the data blocks which are well maintained are verified, this avoids revealing the data corruption. To resist this attack, the challenging messages should not be predetermined by any participant. Existing schemes [17], [18], [19] utilize Bitcoin to generate the challenging messages to ensure the randomness of challenged data blocks, where the auditor extracts the hash value of the latest block from the Bitcoin blockchain, and generates the challenging message according to the security parameter and the extracted hash value. Since in the Bitcoin blockchain the hash value of a block generated at a future time is unpredictable, this ensures that the auditor cannot generate a bias challenging message to deceive the user, and enables the user to efficiently audit the auditor's behavior. However, such mechanism is

vulnerable to a procrastinating auditor. Assuming the agreed verification period is 1 day, and an epoch is 1 month (i.e., 30 days), this means that the auditor checks the outsourced data integrity one time per day, and the user audits the auditor's behaviors one time per month. Normally, the auditor would perform the verification every day and generate a verification report every 30 days. For a procrastinating auditor, it would not perform the verification on the first 29 days, and would perform the verification 30 times on the last day, where the challenging messages in each verification of the first 29 days
can be regenerated in the 30th day. As such, the verification report only reflects the most recent (the 30th day's) state of integrity for the outsourced data. This deviates from the public verification's original target: if the outsourced data is corrupted, the data owner is able to find it within 1 day (i.e., one verification period). To resist the procrastinating auditor, a straightforward solution is to require the user to audit the auditor's behaviours in a random time interval. However, before the user audits the correctness of auditor's behaviours, she/he needs to interact with the auditor to obtain the data that records the auditor's behaviours for the auditing, this sufficiently gives rise to forge the data for the auditor and cloud server. As such, a procrastinating auditor can pass the user's auditing by colluding with the cloud server. Another straightforward solution is to introduce a trusted service provider who provides a time-stamping service [33]. After each verification, the auditor is required to query the time-stamping on the information, which is used to check the data integrity, and is used to be audited by the user to prove the correctness of its behavior. This enables the information to be time-sensitive, and therefore can resist the procrastinating auditor. Nevertheless, the security of such mechanisms rely on the security and reliability of the timestamping service provider, and the provider here becomes a single point of failure. Furthermore, the provider has to bear heavy communication and computation burden in the case of multiple users and auditors. As such, how to resist the procrastinating auditor without introducing any trusted entity is a very challenging problem.

3. Implementation:

Public Verification of Data Integrity The key idea of the public verification technique is that the user (i.e., data owner) splits the data into multiple blocks, computes a signature for each one, and outsources the data blocks as well as corresponding signatures to the cloud server. When the auditor verifies the data integrity, it chooses a random subset of all data blocks (e.g., sample 300 blocks from 10000 ones) and sends the sampled blocks' indexes (as a challenging message) to the cloud server. The cloud server responses with the corresponding proof, the auditor checks the integrity of challenged blocks by verifying the validity of the proof. If the verification passes, the integrity of entire data set is ensured. The key technique used here is aggregated signature [29], which enables the auditor to verify multiple blocks simultaneously without downloading the data. In public verification schemes, after data outsourcing, the user sets a verification period (i.e., the frequency at which the auditor performs the verification). Then the auditor verifies the outsourced data integrity at the corresponding time. In practice, the auditor generates a verification report containing multiple verification results

(corresponding to multiple periods, we call these periods an epoch). If in any period the verification result is "Reject", it means that the data may be corrupted and the auditor needs to inform the user at once. Otherwise, the auditor generates a verification log and provides the user with the log at the end of each epoch. Since the auditor is able to verify the data integrity without the user's participation, the user can assign the auditor to perform the verification with any period as needed. In other words, from the user's perspective, if the outsourced data is corrupted, the longest delay within which she/he needs to find the data corruption should be the verification period. We stress that the frequency at which the auditor checks the data integrity would not be very high in practice, due to the following reasons. First, the auditor serves multiple users simultaneously. If users require the auditor to perform the data integrity verification with a high frequency, e.g., performing the verification every hour, the auditor would bear a heavy communication and computation burden. Furthermore, the higher frequency to perform the data integrity verification, the more costs to employ the auditor. From a pragmatic standpoint, users would not require the auditor to perform data integrity verification with a high frequency in existing scenarios. Second, performing the data integrity verification with a high frequency would also cause heavy verification burden on the cloud server. As pointed out by Armknecht et al. [17], if integrating security mechanisms into existing cloud systems incurs considerable costs on the cloud service providers, most of the providers would not accept liability for the corresponding security guarantees in their Service Level Agreements (SLAs) and only ensure the service availability.

Implementation:

USER INTERFACE DESIGN

This is the primary module of our project. The necessary role for the user is to maneuver login window to user window. This module has created for the protection purpose. During this login page we've got to enter login user id and watchword. It'll check username and watchword is match or not (valid user id and valid password). If we tend to enter any invalid username or watchword we tend to can't enter into login window to user window it'll shows error message. Therefore we tend to square measure preventing from unauthorized user moving into the login window to user window. It'll give an honest security for our project. Therefore server contain user id and watchword server conjointly check the authentication of the user. It well improves the protection and preventing from unauthorized user enters into the network. In our project we tend to square measure victimization JSP for making style. Here we tend to validate the login user and server authentication.

DATA OWNER REQUEST FOR KEY

Here data owner will register and login and request for key to upload the files. With the use of key only he can upload a file in the cloud. Data owner will request for key to the key center.

KEY CENTER GENERATES THE KEY FOR THE OWNER

In this module, key center checks the data owner list or profile; if data owner is a valid person then the key center generates a key for uploading a file. Otherwise it can't generate a key.

DATA OWNER UPLOADS FILE WITH THAT KEY

In this module, data owner will logging in and have to upload some files i.e. to be pdf or a text file. The uploaded file gets encrypted and stored in the database. While uploading a file, key also be stored there.

SEND FILE FOR AUDITING

Uploaded file will be sent to the auditor for checking purpose. In this module separate auditing team will be there for checking and correcting the files. All uploaded files to be sent here for auditing.

AUDITOR CHECKS THE FILE Auditor checks all files uploaded by all data owner with the file key. Auditor can block the file when there is an incorrect file or an incorrect user.

DATA USER REQUEST FOR FILE Here data user will register and login and request for some files uploaded by the data owner. Data user can view the all files uploaded in the database. Files will be viewed as a encrypted text to the data user. They click on request button it will be sent to admin.

ADMIN ACCEPTS THE REQUEST Admin receives notification after getting log in. here there will be the request sent by other data user. If they accept means, key will be sent for download the file. The key will be sent to the requested data user for downloading file with acceptance notification. Otherwise it will be rejected.

ADMIN MAINTAIN THE DATA Here the admin will login and he can view the files uploaded by data owner, and Admin manages all files uploaded by all data owners the file key. Admin maintains the files in the database.

DATA USER DOWNLOAD THE FILE Here the response notification will be received with the key. The file key sent by admin in the backend for downloading the file. When he downloads the file it asks for entering the key. If it is matched it will be downloaded otherwise key will be wrong.

4 CONCLUSION AND FUTURE WORK

The security analysis demonstrates that proposed system provides the strongest security guarantee compared with existing schemes. We have also conducted a comprehensive performance analysis, which demonstrates that has constant communication overhead and is efficient in terms of computation overhead. For the future work, we will investigate how to construct on other blockchain systems. Since the main drawback of proofs of work (PoW) is the energy consumption, constructing on other blockchain systems (e.g., proofs-of-stake-based blockchain systems) can save energy. However, it requires an elaborated design to achieve the same security guarantee while ensuring the high efficiency. This remains an open research issue that should be further explored. We will also investigate how to utilize blockchain technology to enhance cloud storage systems in terms of security, performance, and functionality. As the outsourced data processing (e.g., outsourced computation and searching over encrypted data) has also played an important role in current information age, we will explore the integration of blockchain into existing schemes which should have a deep impact on outsourced data processing.

5. References:

[1] J. Yu, K. Wang, D. Zeng, C. Zhu, and S. Guo, "Privacy-preserving data aggregation computing in cyber-physical social systems," ACM Transactions on Cyber-Physical Systems, vol. 3, no. 1, p. 8, 2018. [2] H. Ren, H. Li, Y. Dai, K. Yang, and X. Lin, "Querying in internet of things with privacy preserving: Challenges, solutions and opportunities," IEEE Network, vol. 32, no. 6, pp. 144–151, 2018. [3] J. Li, H. Ye, W. Wang, W. Lou, Y. T. Hou, J. Liu, and R. Lu, "Efficient and secure outsourcing of differentially private data publication," in Proc. ESORICS, 2018, pp. 187–206. [4] L. Zhong, Q. Wu, J. Xie, J. Li, and B. Qin, "A secure versatile light payment system based on blockchain," Future Generation Computer Systems, vol. 93, pp. 327–337, 2019. [5] G. Xu, H. Li, Y. Dai, K. Yang, and X. Lin, "Enabling efficient and geometric range query with access control over encrypted spatial data," IEEE Trans. Information Forensics and Security, vol. 14, no. 4, pp. 870–885, 2019. [6] K. Yang, K. Zhang, X. Jia, M. A. Hasan, and X. Shen, "Privacypreserving attribute-keyword based data publish-subscribe service on cloud platforms," Information Sciences, vol. 387, pp. 116– 131, 2017. [7] W. Shen, B. Yin, X. Cao, Y. Cheng, and X. Shen, "A distributed secure outsourcing scheme for solving linear algebraic equations in ad hoc clouds," IEEE Trans. Cloud Computing, to appear, doi: 10.1109/TCC.2016.2647718. [8] H. Yang, X. Wang, C. Yang, X. Cong, and Y. Zhang, "Securing content-centric networks with content-based encryption," Journal of Network and Computer Applications, vol. 128, pp. 21–32, 2019. [9] Q. Wang, C. Wang, J. Li, K. Ren, and W. Lou, "Enabling public verifiability and data dynamics for storage security in cloud computing," in Proc. of ESORICS, 2009, pp. 355–370. [10] X. Zhang, H. Wang, and C. Xu, "Identity-based key-exposure resilient cloud storage public auditing scheme from lattices," Information Sciences, vol. 472, pp. 223–234, 2018. [11] K. Wang, J. Yu, X. Liu, and S. Guo, "A pre-authentication approach to proxy re-encryption in big data context," IEEE Transactions on Big Data, 2017, to appear, doi. 10.1109/TBDATA.2017.2702176. [12] J. Ni, K. Zhang, Y. Yu, X. Lin, and X. Shen, "Providing task allocation and secure deduplication for mobile crowdsensing via fog computing," IEEE Transactions on Dependable and Secure Computing, to appear, doi. 10.1109/TDSC.2018.2791432. [13] Y. Zhang, C. Xu, X. Liang, H. Li, Y. Mu, and X. Zhang, "Efficient public verification of data integrity for cloud storage systems from indistinguishability obfuscation," IEEE Trans. Information Forensics and Security, vol. 12, no. 3, pp. 676–688, 2017. [14] H. Shacham and B. Waters, "Compact proofs of retrievability," Journal of Cryptology, vol. 26, no. 3, pp. 442–483, 2013. [15] K. Yang and X. Jia, "An efficient and secure dynamic auditing protocol for data storage in cloud computing," IEEE Trans. Parallel and Distributed Systems, vol. 24, no. 9, pp. 1717–1726, 2013. [16] G. Ateniese, R. Burns, R. Curtmola, J. Herring, L. Kissner, Z. Peterson, and D. Song, "Provable data possession at untrusted stores," in Proc. of ACM CCS, 2007, pp. 598–609. [17] F. Armknecht, J. Bohli, G. O. Karame, and W. Li, "Outsourcing proofs of retrievability," IEEE Trans. Cloud Computing, to appear, doi: 10.1109/TCC.2018.2865554. [18] Y. Zhang, C. Xu, S. Yu, H. Li, and X. Zhang, "SCLPV: Secure certificateless public verification for cloud-based cyber-physicalsocial

systems against malicious auditors," IEEE Trans. Computational Social Systems, vol. 2, no. 4, pp. 159–170, 2015. [19] Y. Zhang, C. Xu, H. Li, and X. Liang, "Cryptographic public verification of data integrity for cloud storage systems," IEEE Cloud Computing, vol. 3, no. 5, pp. 44–52, 2016. [20] J. Sun and Y. Fang, "Cross-domain data sharing in distributed electronic health record systems," IEEE Trans. Parallel and Distributed Systems, vol. 21, no. 6, pp. 754–764, 2010. [21] H. Li, Y. Yang, Y. Dai, J. Bai, S. Yu, and Y. Xiang, "Achieving secure and efficient dynamic searchable symmetric encryption over medical cloud data," IEEE Trans. Cloud Computing, to appear, doi: 10.1109/TCC.2017.2769645. [22] Y. Zhang, C. Xu, H. Li, K. Yang, J. Zhou, and X. Lin, "HealthDep: An efficient and secure deduplication scheme for cloud-assisted ehealth systems," IEEE Trans. Industrial Informatics, vol. 14, no. 9. [23] B. Wang, B. Li, H. Li, and F. Li, "Certificateless public auditing for data integrity in the cloud," in Proc. of IEEE CNS, 2013, pp. 136–144.

[24] D. He, S. Zeadally, and L. Wu, "Certificateless public auditing scheme for cloud-assisted wireless body area networks," IEEE System Journal, vol. 12, no. 1, pp. 64–73, 2018. [25] S. Nakamoto, "Bitcoin: A peer-to-peer electronic cash system." https://bitcoin.org/bitcoin.pdf. [26] G. Wood, "Ethereum: A secure decentralised generalised transaction ledger," Ethereum Project Yellow Paper, vol. 151, 2014. [27] F. Tschorsch and B. Scheuermann, "Bitcoin and beyond: A technical survey on decentralized digital currencies," IEEE Communications Survey & Tutorials, vol. 18, no. 3, pp. 2084–2123, 2016. [28] S. S. Al-Riyami and K. G. Paterson, "Certificateless public key cryptography," in Proc. of ASIACRYPT, 2003, pp. 452–473. [29] D. Boneh, B. Lynn, and H. Shacham, "Short signatures from the weil pairing," in Proc. of ASIACRYPT, 2001, pp. 514–532. [30] A. Juels and B. S. K. Jr, "Pors: Proofs of retrievability for large files," in Proc. of ACM CCS, 2007, pp. 583–597. [31] C. Wang, S. S. Chow, Q. Wang, K. Ren, and W. Lou, "Privacypreserving public auditing for secure cloud storage," IEEE Trans. Computers, vol. 62, no. 2, pp. 362–375, 2013. [32] S. Worku, C. Xu, J. Zhao, and X. He, "Secure and efficient privacypreserving public auditing scheme for cloud storage," Computers & Electrical Engineering, vol. 40, no. 5, pp. 1703–1713, 2014. [33] S. Haber and W. S. Stornetta, "How to time-stamp a digital document," in Proc. of CRYPTO, 1990, pp. 437–455. [34] L. Zhang, B. Qin, Q. Wu, and F. Zhang, "Efficient many-to-one authentication with certificateless aggregate signatures," Computer Network, vol. 54, pp. 2482–2491, 2010. [35] C. Xu, K. Wang, P. Li, S. Guo, J. Luo, B. Ye, and M. Guo, "Making big data open in edges: A resource-efficient blockchain-based approach," IEEE Trans. Parallel and Distributed Systems, to appear, doi: 10.1109/TPDS.2018.2871449. [36] Y. Zhang, X. Lin, and C. Xu, "Blockchain-based secure data provenance for cloud storage," in Proc. ICICS, 2018, pp. 3–19. [37] M. Li, J. Weng, A. Yang, W. Lu, Y. Zhang, L. Hou, J. Liu, Y. Xiang, and R. Deng, "Crowdbc: A blockchain-based decentralized framework for crowdsourcing," IEEE Trans. Parallel and Distributed Systems, to appear, doi: 10.1109/TPDS.2018.2881735. [38] R. C. Merkle, "Protocols for public key cryptosystems," in Proc. of IEEE S & P, 1980, pp. 122–134. [39] J. Garay, A. Kiayias, and N. Leonardos, "The bitcoin backbone protocol: Analysis and applications," in Proc. EUROCRYPT, 2015,

pp. 281–310. [40] A. Kiayias and G. Panagiotakos, "Speed-security tradeoffs in blockchain protocols," IACR Cryptology ePrint Archive, vol. 2015, p. 1019, 2015. [41] R. Goyal and V. Goyal, "Overcoming cryptographic impossibility results using blockchains," in Proc. TCC, 2017, pp. 529–561. [42] C. Pierrot and B. Wesolowski, "Malleability of the blockchains entropy," Cryptography and Communications, vol. 10, no. 1, pp. 211– 233, 2018. [43] J. Katz and Y. Lindell, Introduction to modern cryptography. CRC press, 2014. [44] A. Kiayias, A. Russell, B. David, and R. Oliynykov, "Ouroboros: A provably secure proof-of-stake blockchain protocol," in Proc. CRYPTO, 2017, pp. 357–388. [45] C. Badertscher, U. Maurer, D. Tschudi, and V. Zikas, "Bitcoin as a transaction ledger: A composable treatment," in Proc. CRYPTO, 2017, pp. 324–356. [46] A. Yang, J. Xu, J. Weng, J. Zhou, and D. S. Wong, "Lightweight and privacy-preserving delegatable proofs of storage with data dynamics in cloud storage," IEEE Trans. Cloud Computing, to appear, doi: 10.1109/TCC.2018.2851256. [47] H. Wang, D. He, and S. Tang, "Identity-based proxy-oriented data uploading and remote data integrity checking in public cloud," IEEE Trans. Information Forensics and Security, vol. 11, no. 6, pp. 1165–1176, 2016. [48] M. Sookhak, A. Gani, H. Talebian, A. Akhunzada, S. U. Khan, R. Buyya, and A. Y. Zomaya, "Remote data auditing in cloud computing environments: A survey, taxonomy, and open issues," ACM Computing Surveys, vol. 47, no. 4, pp. 1–34, 2015.