F. Knirsch, A. Unterweger, K. Karlsson, D. Engel, S. B. Wicker,
"Evaluation of a Blockchain-Based Proof-of-Possession Implementation",
Center for Secure Energy Informatics, Salzburg University of Applied Sciences and
School of Electrical and Computer Engineering, Cornell University, Technical Report 2018-01, 2018.

EVALUATION OF A BLOCKCHAIN-BASED PROOF-OF-POSSESSION IMPLEMENTATION

Fabian Knirsch[1], *Andreas Unterweger*[1,2], *Kolbeinn Karlsson*[2], *Dominik Engel*[1], *and Stephen B. Wicker*[2]

[1] Center for Secure Energy Informatics, Salzburg University of Applied Sciences, Puch/Salzburg, Austria
[2] School of Electrical and Computer Engineering, Cornell University, Ithaca, NY, USA

ABSTRACT

Proofs of possession are required to record forensic evidence or to handle copyright claims for images. The use of hashes and signatures, both with and without blockchains, has been proposed for these applications. However, there has been no consideration of the practicality of implementing such schemes on any of the publicly available blockchains. In this work, we describe and evaluate a verifiable, privacy-preserving and computationally binding proof of possession for images using the Ethereum blockchain. We describe an implementation consisting of a *smart contract*, for which we analyze costs of operation and other practical concerns. We find that image size greatly impacts performance and costs, suggesting that rigorous proofs of possession are only feasible on dedicated *private* Ethereum blockchains with modified cost models. These requirements may be eased by providing proofs on compressed or lower-quality copies of the images.

***Index Terms*—** Image, Proof of possession, Blockchain, Smart contract, Ethereum, JPEG, WebP, Costs

1. INTRODUCTION

Cryptographic and perceptual hash functions are commonly used in conjunction with digital signatures to verify the integrity of images [1, 2, 3]. This allows the sender to store or transmit an image file over unsecured channels and the receiver to verify that the file and its contents have not been changed or – in the case of perceptual hashes – have not been changed significantly. While such approaches can verify the integrity of the image, they cannot be used to prove that a person is in possession of the image; the use of replay attack would be trivial, as one need simply resend the signature. Proving that the sender was in possession of the image at a certain point in time remains a challenging problem.

In this paper, we present an approach for verifiable, privacy-preserving and computationally binding proofs of possession based on established cryptographic primitives. A computationally binding and hiding commitment scheme is combined with blockchain technology through the use of a *smart contract*. The contract is public, allowing for verification and trusted execution by third parties due to the properties of the blockchain. The image in question does not need to be revealed until requested. It can even be revealed on a controlled, non-public basis in cases where privacy is required, e.g., police investigations

A two-stage protocol based on the aforementioned building blocks is proposed: The first stage enables proof of possession (PoP); one can prove that a person was in possession of an image at a certain point in time. The second stage builds on the first, additionally enabling supported proof of possession (SPoP) through which others can prove that one was in possession of an image by building on a k-out-of-n threshold scheme of trusted signatures. The protocol can be applied, for example, to the resolution of copyright claims and persisting forensic evidence. Furthermore, the protocol can serve as the basis for a chain of custody for forensic evidences [4].

The contribution of this paper is twofold: First, the blockchain-based image verification scheme is presented as a protocol for PoP and SPoP. Second, a prototypical implementation of the corresponding smart contract in the Ethereum blockchain is created and evaluated. Results in terms of both practicality and costs are discussed for this technology.

The rest of the paper is structured as follows: Section 2 introduces the preliminaries and related work. Sections 3 and 4 present the first and second stages of the scheme for proving the possession of an image, respectively. Section 5 evaluates both schemes based on a prototypical implementation in the Ethereum blockchain. Section 6 summarizes our findings.

2. RELATED WORK

This section presents preliminaries and related work in the field of proof of possession, blockchain technology, smart contracts and commitment schemes.

2.1. Proof of Possession

Proof of possession (PoP), also referred to as *Provable Data Possession* [5, 6], allows an entity to prove that it was in possession of particular data, e.g., an image, at a point in time.

An approach that supports PoP for data stored on an untrusted server is presented by [5] and [6]. These approaches

The financial support by the Federal State of Salzburg is gratefully acknowledged. Funding by the Austrian Research Promotion Agency (FFG) under project number 865082 (ProChain) is gratefully acknowledged.

allow retrieving the full data from a server along with a proof that the data has not been tampered with, but only provide probabilistic security guarantees.

Approaches that incorporate a blockchain instead of a single untrusted server are presented in [7] and [8]. [7] propose the use of smart contracts in Ethereum, which allow for more fine-grain access control than ours, but they do neither evaluate costs nor performance. Similarly, [8] do not perform any practical evaluations, with their protocol supposedly being based on a custom-designed blockchain. In contrast, we evaluate our PoP protocol in the widely adopted Ethereum blockchain and perform a cost analysis with 1000 images.

2.2. Blockchains and Smart Contracts

Blockchain technology was originally conceived of as the underpinnings of the cryptocurrency *Bitcoin* [9], a trust-less, decentralized ledger for financial transactions. Instead of relying on a single trusted third party, a distributed ledger is maintained by all nodes, i.e., participants in the network. Transactions are validated and combined into a block that is permanently and immutably bound to its predecessor, thereby creating a list of blocks linked together by their cryptographic hashes. New blocks can be appended by any node, but they require a proof that a considerable amount of work has been spent by finding a specific output for a deterministic one-way function with a uniformly distributed output. This is referred to as *mining* and provides the basis of the proof-of-work consensus algorithm that – if at least a significant portion of the computing power is spent honestly – ensures the validity and immutability of transactions [10]. Furthermore, transactions are signed in order to verify the sender. A special type of transactions requires multiple signatures, often implemented as a (k, n) threshold scheme. In order to initiate a transaction, $k \leq n$ out of (a predefined list of) n participants must sign it.

Blockchain technology has been proposed for other application domains, such as energy management [11, 12, 13] and the Internet of Things [14]. Blockchains like Ethereum allow more general Turing-complete operations (for all practical purposes) [15]. Such operations can be grouped into methods that form class-like compounds with additional state information. These compounds expose an API which is referred to as a *smart contract*, of which multiple instances can be created. The state of each instance of a smart contract (as well as the code of the smart contract itself) is publicly available and stored in the blockchain, as is the result of every computation permuting its state. The correctness of the computations and their results can be verified by all nodes.

In summary, the main features of blockchain technology, including Ethereum, that are relevant for this paper are:

- **Decentralization:** Instead of relying on a single trusted entity, trust is spread across multiple participants, depending on the agreed-upon consensus algorithm [16].
- **Immutability**: Once a transaction is committed to the blockchain and a sufficient number of participants have agreed on the state, it is persisted practically immutably.
- **Limited Privacy**: All data in the blockchain is publicly visible to all participants. In order to achieve privacy, additional layers, e.g., commitment schemes, are required [13].

2.3. Commitment Schemes

A computationally binding and hiding commitment scheme $C = \{\mathrm{commit}, \mathrm{open}\}$ is proposed in [12, 13, 17]. In order to commit to a value, e.g., an image I, $c = \mathrm{commit}_H(I, r) = H(I|r)$ is computed using a random number r (the secret) and a cryptographic hash function H, with the $|$ symbol denoting concatenation. The resulting value c is stored as part of the state of a smart contract, for example in the case of Ethereum.

In order to later verify the commitment, I and r need to be provided by the original committer so that $\mathrm{open}_H(I, r, c) = (c' = H(I|r), (c = c' \rightarrow true) \wedge (c \neq c' \rightarrow false))$ can be computed. If $c = c'$, I has not been changed and thus the commitment is verified and open returns *true*. In the opposite case, either I or r have been tampered with or changed since committing, making the commitment invalid and thus open returns *false*. If H is hard to invert (which cryptographic hash functions are by design), producing an image which yields the same hash is impractically expensive.

The hash function and the immutability of the data in the blockchain ensure the hiding and the binding property for both the value I and the hash c. The latter therefore allows to not only check the integrity of the committed value, but also to rely on the integrity of the commitment itself.

3. PROOF OF POSSESSION

The first stage of the protocol allows one to prove possession by *committing* to an image, i.e., providing the hash of the image, and to later *open* the commitment to prove the previous possession as well as the integrity of this image. The proposed scheme is based on computationally binding commitments and blockchain technology. It consists of three phases named (i) preparation, (ii) commitment, and (iii) opening. The scheme achieves the following properties:

- **Commitment**: A commitment to an image at a certain point in time to prove that the committer was in possession of an untampered version of the image.
- **Privacy preservation**: A commitment to an image without revealing the actual image at the time of committing.
- **Integrity**: Proof of the image's integrity after storage or transmission by opening the commitment. This implies relying on a decentralized and trust-less architecture for the integrity of the hash.
- **Verifiability**: Public verifiability by anyone when a commitment has been made, opened and verified at least once.

In the following, each phase of the process is described in detail. Consider the following use case: Alice wants to commit to an untampered version of an image at some point in time and later pass the image to Bob, along with a proof of the integrity of that image as well as a proof that Alice was in possession of the image at the time of the commitment.

Given an image I, Alice first draws a fresh random number r from a cryptographically secure random number generator and computes a hash $c = H(I|r)$ for this image and the random number, as described above. In order to sign her transactions in the blockchain, Alice further establishes a public-private key pair $(\mathrm{pk}, \mathrm{sk})$.

Alice establishes a new instance of the smart contract by calling the constructor and passing the hash c as an argument, which is stored in the blockchain by the smart contract. The corresponding unique address a of the contract instance (provided by the network) is required for further interactions. The transaction performed to create the smart contract is signed with Alice's private key sk. Once the transaction is mined and stored permanently in the blockchain, everyone can verify that (i) a commitment has been stored by Alice and (ii) that the committed hash has not been changed due to the immutability of data in the blockchain.

Alice's real-world identity can be connected to her public key using a web of trust [18]. A web of trust links identities to keys with a chain of mutually trusted intermediaries. For example, if Bob vets Charlie's identity, and Charlie vets Alice's identity, then Bob considers Alice's identity vetted.

Alice now passes the image to Bob, along with the address a of the smart contract and the random number r. The latter allows Bob to verify that the commitment has actually been sent to the blockchain by Alice. Bob receives (I, r, a).

Given a, Bob is able to retrieve the instance of the smart contract and to call the open method with the arguments I and r. This call goes to the same instance a of the smart contract that has been used by Alice in the commitment phase. The smart contract recomputes $H(I|r) = c'$ and checks whether c' equals the previously stored c. If the integrity of the image is verified, the smart contract outputs *true*, otherwise *false*. For privacy-sensitive use cases, the check can be performed offline by calculating c' and comparing it to c without invoking the smart contract. In this case, however, there is no record that I existed, its integrity has been checked, or that Bob was in possession of the image for verification.

4. SUPPORTED PROOF OF POSSESSION

This stage builds on the first stage as presented in Section 3, but additionally allows others to support, i.e., notarize, the proof of possession of an image. For this scheme, an additional support phase is needed.

The preparation phase is the same as in the PoP scheme with the small change that Alice calls the constructor that additionally accepts n public keys, $p = \{\mathrm{pk}_1, \mathrm{pk}_2, \ldots, \mathrm{pk}_n\}$, that can support the proof of possession and a threshold $k \leq n$. In order to prevent Sybil attacks, the identities behind those keys are verified via the aforementioned web of trust.

Any participants whose public key has been listed in the SPoP smart contract instance can call the sign method after construction. Calling a method in a smart contract implicitly passes the public key of the caller $\mathrm{pk}_{\mathrm{caller}}$ as an argument and implicitly signs the method call and transactions, respectively. The smart contract checks whether $\mathrm{pk}_c \in p$. If it is, the smart contract increments the number of supporting participants.

The opening phase is similar to the one in PoP. While Bob still uses (I, r, a) to open the commitment, the smart contract does not only recompute $H(I|r) = c'$ and check with the previously stored c whether $c = c'$, but also checks whether at least k out of n supporters have called the sign method.

In summary, the PoP and SPoP schemes have the following privacy and integrity features: Both schemes enable proving possession of I at the time of committing. If the smart contract outputs *true* after opening, Bob knows that Alice was in possession of I at the time of creating the smart contract instance and Bob knows that I has not been tampered with. Furthermore, until Alice releases I, the hiding feature of the commitment scheme prevents others from learning anything about the image. Finally, by calling the smart contract in the opening phase, Alice knows that Bob verified the integrity of the image. The SPoP scheme extends these abilities with an additional support from third parties through their signatures.

5. EVALUATION

For evaluation, we implemented the PoP and SPoP smart contracts for Ethereum in Solidity[1] with compiler optimization enabled. These smart contracts offer the following methods that reflect the semantics described in the previous section:

- `ProofOfPossession(c)`. This constructor establishes a new instance of the smart contract for PoP and stores a pre-computed hash c in the blockchain.
- `SupportedProofOfPossession(c, k, pk`$_1$`, pk`$_2$`, ..., pk`$_n$`)`. This constructor establishes a new instance of the smart contract for SPoP, stores a pre-computed hash c in the blockchain, and accepts a list of public keys (represented by addresses in Ethereum) of n participants, as well as a threshold k.
- `sign()`. For SPoP only, this method can be called by any of the participants specified in the instance's constructor. By calling it, the caller notarizes the proof of possession. Since calling a method is a transaction, an implicit signature is recorded permanently in the blockchain.
- `open(I, r)`. Calculates c' given I and r, and verifies whether $c = c'$. For SPoP only, this method additionally checks if k or more out of n participants have signed. If so, it returns *true* if and only if $c = c'$, and *false* otherwise.

To evaluate our smart contracts, we set up a local blockchain using *web3.py*[2] 3.0 and its dependencies.

[1]http://solidity.readthedocs.io/en/develop/
[2]https://github.com/ethereum/web3.py

Operations from smart contract code are executed in the Ethereum Virtual Machine (EVM). To prevent the network from being flooded with invalid requests, each EVM operation induces costs to be paid by the caller. Cost is measured in *gas* [15] and corresponds to prices in the cryptocurrency Ether, depending on the time of execution and other factors [19]. The local blockchain that we use for our evaluation does not require spending actual Ether, but still allows for very accurate estimates of the costs that would be induced in the public Ethereum network since the same EVM is used.

We use the reference images of the Tampere Image Database 2013 [20] for our evaluation. We compress them using the JPEG and WebP encoders of ImageMagick[3] 6.8.9-9 with a step size of 5% within the range of valid values of the quality parameter (0-100% for JPEG, 5-100 for WebP).

5.1. Costs

Fig. 1 illustrates the costs for deploying the PoP smart contract and executing its methods for the minimum number of times in the order specified above. It is clear that the (file) size of the images (x axis) impacts the costs (y axis) linearly (approx. EUR 1.50/KiB) for all practical purposes. Thus, WebP images compressed with the same quality parameter as the corresponding JPEG images incur less costs. However, while compressing images saves space and costs, this might undermine the PoP. The extent to which lower-quality copies of images can be used as evidence is out of scope for this paper and needs to be carefully evaluated for each use case. Overall, the costs of several hundreds of Euros for images of higher quality (and thus larger file size) confirm previous findings from [13] showing that handling data of several kilobytes in size is impractically expensive for many use cases, including ours.

The bottom-right of Fig. 1 shows an enlarged view of the smallest, i.e., lowest-quality, images. Two effects are apparent: First, the constant size of the smart contract (code) as well as the constant base cost per method call [15] define a lower bound for the total cost at about five Euros at the time of writing. Second, small WebP images cost more to deploy than JPEG images of the same size. This is due to the fact that, due to their Huffman tables, JPEG images have more zero bytes, which cost less to pass as parameters than non-zero bytes, resulting in slightly decreased total costs [15]. For larger images, this effect disappears due to the smaller relative size of the tables compared to the coded image data.

The costs for the SPoP smart contract are practically identical to the ones for the PoP contract. The slightly more complex code yields a bigger code size for the deployment as well as a higher number of method calls (k additional `sign` calls) and additional checks. The combination of both results in higher minimum costs (about double) compared to PoP, which is likely negligible for practical quality levels, i.e., larger image sizes, which induce high overall costs.

[3]https://www.imagemagick.org/script/index.php

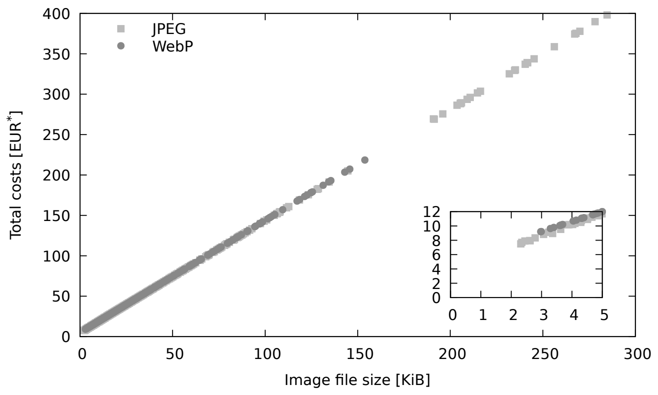

Fig. 1. Estimated cost for executing the PoP smart contract on the Ethereum blockchain for images of different (file) sizes. * The price of Ether is approximated by 1000 Euros per ETH reflecting its average at the time of writing (January 2018).

5.2. Practical concerns

During our evaluation, we made two additional observations. First, the public Ethereum blockchain imposes a gas limit per transaction. This means that, at the time of writing (at a limit[4] of $8 \cdot 10^6$gas $\approx$ EUR 160), the maximum image size is limited to about 100 KiB (cf. Fig. 1). Creating a custom (so-called private) blockchain with different limits and/or a different cost structure among a select group of participants can mitigate this problem. Second, the processing time of image data is around 0.1 s/KiB for both smart contracts. This is due to the EVM processing parameters on a machine-word (256-bit) basis, with each store operation updating a modified Merkle Patricia Trie [15], which is relatively time-consuming. Our local evaluation did not include any network delays, meaning that processing on the public Ethereum blockchain will take significantly longer. It is likely that blockchains with different storage and cost models are more suitable for cost-effective execution of PoP and SPoP smart contracts.

6. CONCLUSION

We described and evaluated two smart contracts for proving image possession through commitments. We showed that the image size impacts costs linearly in practice and that the total costs exceed 150 Euros for images of 100 KiB size. If the commitment is not opened publicly, only a small fraction of these costs incur. This comes, however, at the cost of less rigorous guarantees for integrity. Due to the specific design of Ethereum, we therefore suggest using the proposed smart contracts only on a private blockchain or recommend switching to a different blockchain optimized for storage. Future work will focus on evaluating the ability of smart contracts to serve as a chain of custody for forensic evidence.

[4]https://ethgasstation.info/

7. REFERENCES

[1] Jiri Fridrich and Miroslav Goljan, "Robust hash functions for digital watermarking," in *Proceedings International Conference on Information Technology: Coding and Computing (Cat. No.PR00540)*, 2000, pp. 178–183.

[2] Vishal Monga, Arindam Banerjee, and Brian L. Evans, "A Clustering Based Approach to Perceptual Image Hashing," *IEEE Transactions on Information Forensics and Security*, vol. 1, no. 1, pp. 68–79, 2006.

[3] Ching-yung Lin and Shih-fu Chang, "Generating Robust Digital Signature for Image / Video Authentication," in *Multimedia and Security Workshop at ACM Multimedia '98*, Bristol, U.K., 1998.

[4] Simson L. Garfinkel, "Providing Cryptographic Security and Evidentiary Chain-of-Custody with the Advanced Forensic Format, Library, and Tools," *International Journal of Digital Crime and Forensics (IJDCF)*, vol. 1, no. 1, pp. 1–28, 2009.

[5] Giuseppe Ateniese, Randal Burns, Reza Curtmola, Joseph Herring, Lea Kissner, Zachary Peterson, and Dawn Song, "Provable Data Possession at Untrusted Stores," *Proceedings of the 14th ACM conference on Computer and communications security - CCS '07*, pp. 598–609, 2007.

[6] Chris Erway, Alptekin Küpçü, Charalampos Papamanthou, and Roberto Tamassia, "Dynamic Provable Data Possession," *ACM Transactions on Information and System Security*, vol. 17, no. 4, pp. 15:1–15:29, apr 2015.

[7] Ricardo Neisse, Gary Steri, and Igor Nai-Fovino, "A Blockchain-based Approach for Data Accountability and Provenance Tracking," in *12th International Conference on Availability, Reliability and Security (ARES)*, Reggio Calabria, Italy, 2017, ACM.

[8] Igor Zikratov, Alexander Kuzmin, Vladislav Akimenko, Viktor Niculichev, and Lucas Yalansky, "Ensuring Data Integrity Using Blockchain Technology," *20th Conference of Open Innovations Association (FRUCT)*, pp. 534–539, 2017.

[9] Satoshi Nakamoto, "Bitcoin: A Peer-to-Peer Electronic Cash System," Tech. Rep., 2008.

[10] Florian Tschorsch and Björn Scheuermann, "Bitcoin and Beyond: A Technical Survey on Decentralized Digital Currencies," *IEEE Communications Surveys & Tutorials*, vol. 18, no. 3, pp. 2084–2123, 2016.

[11] Fabian Knirsch, Andreas Unterweger, Günther Eibl, and Dominik Engel, "Privacy-Preserving Smart Grid Tariff Decisions with Blockchain-Based Smart Contracts," in *Sustainable Cloud and Energy Services: Principles and Practices*, Wilson Rivera, Ed., chapter 4, pp. 85–116. Springer International Publishing, 2017.

[12] Fabian Knirsch, Andreas Unterweger, and Dominik Engel, "Privacy-preserving Blockchain-based Electric Vehicle Charging with Dynamic Tariff Decisions," *Journal on Computer Science - Research and Development (CSRD)*, vol. 33, no. 1, pp. 71–79, feb 2018.

[13] Andreas Unterweger, Fabian Knirsch, Christoph Leixnering, and Dominik Engel, "Lessons Learned from Implementing a Privacy-Preserving Smart Contract in Ethereum," in *2018 9th IFIP International Conference on New Technologies, Mobility and Security (NTMS)*, accepted.

[14] Konstantinos Christidis and Michael Devetsikiotis, "Blockchains and Smart Contracts for the Internet of Things," *IEEE Access*, vol. 4, pp. 2292–2303, 2016.

[15] Gavin Wood, "Ethereum: A Secure Decentralised Generalised Transaction Ledger," Tech. Rep., Ethereum, 2017.

[16] Kyle Croman, Christian Decker, Ittay Eyal, Adem Efe Gencer, Ari Juels, Ahmed Kosba, Andrew Miller, Prateek Saxena, Elaine Shi, Emin Gün Sirer, Dawn Song, and Roger Wattenhofer, "On Scaling Decentralized Blockchains," in *Financial Cryptography and Data Security, Christ Church, Barbados, February 26, 2016, Revised Selected Papers*, Jeremy Clark, Sarah Meiklejohn, Peter Y.A. Ryan, Dan Wallach, Michael Brenner, and Kurt Rohloff, Eds., pp. 106–125. Springer, Berlin Heidelberg, 2016.

[17] Kevin Delmolino, Mitchell Arnett, Ahmed E Kosba, Andrew Miller, and Elaine Shi, "Step by Step Towards Creating a Safe Smart Contract: Lessons and Insights from a Cryptocurrency Lab.," in *Financial Cryptography and Data Security*, Barbados, 2016, pp. 79–94, International Financial Cryptography Association.

[18] Simon Garfinkel, *PGP: Pretty Good Privacy*, O'Reilly Media, 1994.

[19] Chris Dannen, *Introducing Ethereum and Solidity*, Apress, 2017.

[20] Nikolay Ponomarenko, Lina Jin, Oleg Ieremeiev, Vladimir Lukin, Karen Egiazarian, Jaakko Astola, Benoit Vozel, Kacem Chehdi, Marco Carli, Federica Battisti, and C. C. Jay Kuo, "Image database TID2013: Peculiarities, results and perspectives," *Signal Processing: Image Communication*, vol. 30, pp. 57–77, 2015.