Towards Distributed Trustworthy Traceability and Accountability

Jörn Erbguth [a] and Jean-Henry Morin [a b]

[a] *University of Geneva, CUI - ISS, 1227 Carouge, Switzerland*
Tel: +41 787256027, E-mail: erbguth@unige.ch - Tel: +41 22 379 02 55, E-mail: Jean-Henry.Morin@unige.ch

[b] *Korea University Business School, Seoul, South Korea*
Tel: +82 2 3290 28 93, E-mail: morinj@korea.ac.kr

Abstract

Digital traces play an increasingly important role in our society. Whether in the context of regulatory compliance, contractual exchanges or simply for general interactions, people need to be able to document trustworthy facts. Most approaches today rely either on Trusted Third Parties, at best, or more generally on collecting such traces after problems occur in ways where their authenticity may be arguable (fabricated, doctored). Blockchain technology offers an interesting alternative to the problem by allowing documenting transactions in a distributed consensus ledger with transparency and immutability properties. This paper proposes a new approach to the problem leveraging blockchain technology towards providing a framework for distributed trustworthy logging of digital facts and traces on the blockchain as they happen or are needed before problems arise. Disintermediation of such processes is likely to significantly help raise trust and accountability in many aspects of our interactions, whether online or offline.

Keywords:

Compliance, Distributed Trust, Blockchain, Logging, Digital Traces, Proof

Ⅰ. Introduction

Our society is increasingly relying on digital services and interactions. Most of the time things go well and little attention is paid to anticipating potential problems. However, when problems arise, we often wish we had been able to rely on some form of authoritative traces to prove our case. Such situations frequently lead to a digital quest trying to dig up electronic bits and pieces of information to provide as evidence to support our claims. This paper proposes to revisit this problem by looking at how blockchain technology can help better prepare for such situations by providing a simple approach allowing to log digital traces and facts in a decentralized and trustworthy way. The next section describes the problem and the requirements that should be met to achieve this goal. Section three presents and argues why blockchain technology is a key element to address the problem before proposing a design for a framework we called BlockTrace. We then discuss related work and existing approaches before concluding.

Ⅱ. Problem Statement and Requirements

Service request and provisioning involve exchanging messages. Although all systems maintain logs they essentially remain locked in silos and rarely carry any form of publicly verifiable accountability. Worse, logging is often more an issue of internal readiness to face problems than regular preparedness in regular operations. As a result, the issue that needs to be addressed is: how might we be able to provide a way for services and people to simply document digital traces in a publicly accountable and trustworthy way without relying on trusted third parties. In other words, can we design something allowing systems, services and people alike to be re-empowered in their digital responsibility preparedness level before problems occur rather than facing the current digital haystack of untrustworthy traces and evidence that need to be collected after problems arise?

Today, digital traces to be produced as evidence can be easily fabricated or doctored, hence the growing need for digital forensics. We still largely rely on contextual probabilities where undisputable proofs would be desirable. Software, services and users have almost no option or choice whether to generate digital traces and no power in selecting the traces to be presented in case of dispute. Services that provide secure proof, such as timestamping, are good examples of notarized services but rely on trusted third parties, are often cumbersome to use and are rarely integrated in with common software or services on the side of their users.

To address this issue we need to find a way allowing the easy recording of trustworthy digital traces for users and service developers alike. A key requirement in this context is not to have to rely on a centralized trusted third party. Equally important is the ability to log and verify digital traces asynchronously on a publicly accessible repository. Trust, accountability and security are of utmost importance for such an approach. Therefore, recorded traces must be persistent, immutable and privacy preserving when necessary. Finally, in order for such an approach to be generalizable, and therefor useful, it should be considered as an open framework allowing for a variety of technologies to be used (e.g., cryptographic algorithms).

Ⅲ. A Distributed Trust Approach Based on Blockchain

To this end and to meet the above requirements, the now more than emerging blockchain technology appears to provide some the needed fundamental properties. First and foremost, blockchain technology is a distributed ledger allowing to record transactions with three major characteristics. Transparency: all transactions written on the blockchain are visible to everyone. Persistency and immutability: transactions are collected in blocks linked to one another through cryptographic hash functions. As a result, they cannot be changed without invalidating the hashes. The blockchain is basically replicated in whole at all the nodes of the blockchain distributed network. The consensus is achieved through distributed consensus algorithms, thus providing a distributed trust network much more reliable and accountable than centralized trusted third party approaches.

Therefore, blockchain technology [1] exhibits many of the fundamental properties needed to achieve our goal. The rest of this section presents "BlockTrace", a tentative design towards a framework for distributed trust logging of digital traces based on blockchain technology.

The proposed framework is based on asymmetric and symmetric cryptography, and one-way hash functions [2] to ensure the desired level of privacy as well as blockchain technology to meet the design requirements.

From a high level point of view, the approach can be described in four layers (Figure 1). The base layer upon which our design sits is the blockchain layer serving as storage layer. The second layer is the proposed BlockTrace framework enabling the management of traces together with the corresponding metadata. The third layer is a trace management layer allowing the organization of traces into trails of connected traces. It also allows managing the different cryptographic keys and hashing functions. Every trace can use a different encryption key for security and privacy reasons allowing trace isolation. The top layer is basically the application layer using the framework such as for example compliance, contracts or documents.

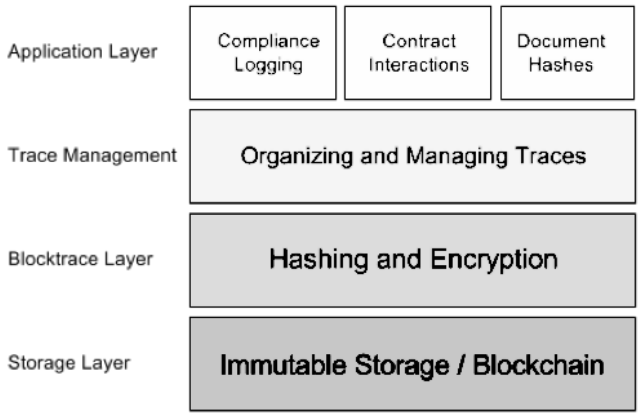

Figure 1 – Layers of the Trace Recording

The storage layer, basically any blockchain infrastructure, records a payload together with some blockchain dependent metadata as a blockchain transaction. Usually this metadata consists of a reference to the smart contract that is addressed for the storage, the sender who is paying for the transaction and through the block identification the approximate time. On top of the storage layer, the BlockTrace layer records a hash of the content for which a trace is needed (Trace Content Hash) and some contextual information (Trace Context). The context can be used to identify what was hashed, a related piece of information, basically anything making sense in relation to the trace to be logged.. An optional trace signature may also be added to the payload. Figure 2 shows the overall structure of the BlockTrace transaction in the blockchain layer.

Example use cases may cover automated logging from services and software applications, contractual interactions in any form, including from a web page or even screen captures. We anticipate many more use cases to be further defined and documented but this isn't our focus here.

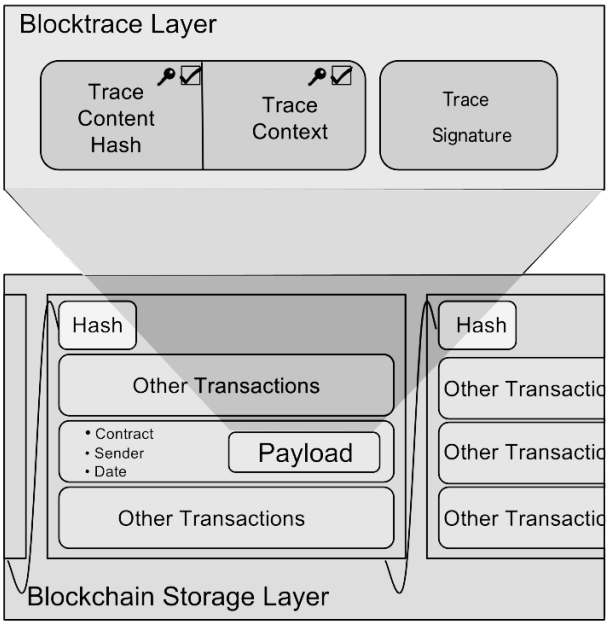

Figure 2 – Blockchain and Blocktrace Layer

From the framework point of view, the Blocktrace layer will provide an API to write (putTrace) and read (getTrace) traces. The method to write a trace is presented in figure 3. The information that is to be traced is transmitted as a file **traceContent**. It is hashed locally in order to avoid sending private information. The **traceContext** is the metadata, claims or other related information an application or user wants to link to the trace. Since it will be written on the blockchain its length needs to be minimized. An encryption of the hash of the **traceContent** as well as the **traceContext** is possible together with a choice of the encryption methods. With the **traceSignature** a trace can be authenticated. The **sender** is the blockchain account paying for the transaction fee. The password for the account might be provided by a configuration or a callback.

Registering long records on the blockchain will result in high transaction fees. This will be the case, if the

traceContext is long. By default, a blockchain dependent transaction-fee-limit will be set. This can be overwritten using the parameter **txFeeLimit**. With the flag **critical** the system is advised to do everything it can to get a trace written to the blockchain. The **txFeeLimit** will be ignored and unsuccessful writing of transactions can be repeated with higher fuel limits if necessary.

putTrace (

- traceContent — file
- traceContext — string
- traceSignature — string, optional
- hashEncryption — boolean, optional
- contextEncryption — boolean, optional
- encryptionAlgorithm — enum, optional
- encryptionKey — string, optional
- sender — blockchain-id
- txFeeLimit — int, optional
- critical — boolean, optional **)**

Figure 3 – Parameters of the trace writing method

Since writing on the blockchain is asynchronous the success of writing a trace including the transactionId can be retrieved through a callback.
For querying the blockchain we also propose a sample method (figure 4). Since the blockchain is public, the information could also be accessed directly. However, the data structure on the blockchain might change with versions of the contract while the API can be kept stable.

getTrace (

- transactionId
- traceContext – if no encryption used
- enc(traceContext) – if encryption used
- hash(traceContent) – if no encryption used
- enc(hash(traceContent)) – if encryption used
- sender
- date/time-range **)**

Figure 4 – Alternative parameters for retrieving a trace

The trace context as well as the trace hash can be encrypted. Whether encryption is useful or required depends on the use case. There are four different possible encryption policies depending on whether the hash trace and context are encrypted or not (Table 1).

Table 1 – Encryption Scenarios

Case	Hash	Context	Meaning
a)	Plain text	Plain text	Log readable and content verifiable for all
b)	Plain text	Encrypted	Content verifiable but without context
c)	Encrypted	Plain text	Log readable but content only verifiable with consent of the key holder
d)	Encrypted	Encrypted	Completely private traces

IV. Related Work

The most relevant related work is in the area of time stamping services. A time stamping protocol exists since 2001 (RFC 3161) [3]. However, this protocol requires trust in a Time Stamp Authority (TSA) that signs the document with a trustworthy time value. Two trusted party are involved: one providing the time and a second verifying the digital signature of the time provider. A certification authority provides the key used. The validity of these signatures are limited and the time stamping has to be repeated before the signature expires.
Another example is the *icanprove* service [4] that provides a virtual browser session in which a user can navigate a website and take virtual screen-shots at any time. The screen-shots together with metadata of the session are documented in a signed pdf by the service. This service is a very specific use case, relying on trust in that service and signatures need regular resigning. When tried the PDF did not contain a signature. A specific application of time-stamping in medical research is proposed in [5].
To experiment with blockchain technology in this context we have implemented very easily such a time-stamping service for file tracing. This rough prototype was implemented on Ethereum in less than a day. Similar services are BTProof [6] and OriginStamp [7] [8] for Bitcoin Trusted Time-stamping.

V. Conclusion and Future Work

Time-stamping is a well-established technique to preserve proofs. However conventional time-stamping services require trusted third parties and constant resigning of signed data. Generalizing the issue in terms of trusted traces is a much-needed feature for our digital society. Blockchain technology offers a new perspective in this context providing distributed trustworthy traceability and accountability. We have proposed a preliminary design that needs further development and work.

References

[1] Buterin, V. (2014). A next-generation smart contract and decentralized application platform. *white paper*.
[2] Schneier, B. (2007). *Applied cryptography: protocols, algorithms, and source code in C*. john wiley & sons.
[3] Adams, C., & Pinkas, D. (2001). Internet X. 509 Public Key Infrastructure Time Stamp Protocol (TSP).
[4] www.icanprove.de (retrieved Oct. 4, 2016)
[5] Irving, G., & Holden, J. (2016). How blockchain-timestamped protocols could improve the trustworthiness of medical science. *F1000Research*, *5*.
[6] https://btproof.com (retrieved Oct. 4, 2016)
[7] https://www.originstamp.org (retrieved Oct. 4, 2016)
[8] Gipp, B., Meuschke, N., & Gernandt, A. (2015). Decentralized Trusted Timestamping using the Crypto Currency Bitcoin. *arXiv preprint arXiv:1502.04015*.