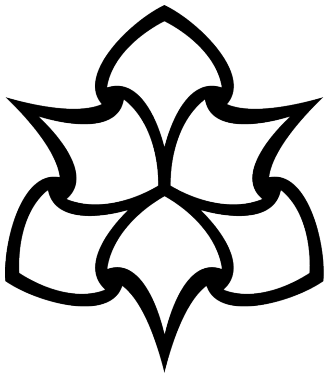

Data Preservation System using BoCA: Blockchain-of-Custody Application

Thomas Martin
Mohammad Hammoudeh
t.martin@mmu.ac.uk
m.hammoudeh@mmu.ac.uk
Department of Computing and Mathematics, Manchester Metropolitan University
Manchester, UK

ABSTRACT

The recent growth in popularity of blockchains has been primarily fueled by their support of cryptocurrencies and smart contracts. However, a blockchain is primarily a distributed, trustworthy ledger and one of its greatest features is its ability to preserve data in a robust and verifiable manner. In this paper we present a general Blockchain-of-Custody Application proof-of-concept. BoCA allows data to be preserved on the blockchain in a transparent and verifiable way. We present our application and discuss the advantages this approach has over the state-of-the-art, as well as discuss different scenarios in which it could be used.

CCS CONCEPTS

• **Security and privacy** → *Cryptography*; • **Applied computing** → *Computer forensics.*

KEYWORDS

blockchain, data preservation, data integrity, digital forensics, chain of custody

ACM Reference Format:
Thomas Martin and Mohammad Hammoudeh. 2021. Data Preservation System using BoCA: Blockchain-of-Custody Application. In *The 5th International Conference on Future Networks & Distributed Systems (ICFNDS 2021), December 15–16, 2021, Dubai, United Arab Emirates.* ACM, New York, NY, USA, 8 pages. https://doi.org/10.1145/3508072.3508084

Information is intangible, which often makes it ephemeral. Whereas physical objects can be held and contained, information can often be duplicated and modified at will. At the same time, the blurring of the distinction between our physical lives and our digital lives makes the reliability and integrity of our information critically important. To create a system that gives the utmost assurances of protection of information, we should look to a disciple where integrity is fundamental: Digital Forensics. In order to collect and examine digital evidence in a manner than is reliable enough to present in a court of law, one must be able to provide strong guarantees of the integrity of the evidence. Many tools exist to help achieve this aim, with physical documentation, hashes, and digital signatures being the primary methods. While each of these do have their merits, they are not perfect. Handwritten signatures provide only very weak assurances, as they can be forged and are not dependent on the document signed. Hashes do not solve the problem, merely change it from "protect the integrity of the evidence" to "protect the integrity of the hash digest" (granted this is an easier task). Digital signatures do provide strong claims on the author of any statement, but cannot (by themselves) provide strong claims as to when any statement was made, nor can they rule out the possibility of data being erased nor of multiple (conflicting) versions of a document/statement being issued and signed.

In this paper we present a new tool that can provide much greater assurances on the integrity of information. The tool is sufficiently reliable that it could be used to protect evidence to the highest demands required in digital forensics, but has been designed to be suitable for general use. It also makes novel use of the underlying blockchain platform in a way that provides distinct benefits. The Blockchain-of-Custody Application[1] allows a user to timestamp data on a blockchain in a reliable and verifiable way.

The layout of the paper is as follows: Section 1 describes previous related work in this area. Section 2 states the requirements gathered from identifying gaps in the state-of-the-art. Section 3 explains how BoCA works. Section 4 describes the implementation of BoCA with examples. Sections 6 and 7 complete the paper by exploring possibilities for future work and various scenarios where BoCA could be suitably used.

1 PREVIOUS WORK

The concept of the blockchain was first presented by Nakamoto in [16]. It was used to record a verifiably immutable, distributed ledger of transactions for the cryptocurrency Bitcoin.

There are already services that provide the feature of timestamping documents on blockchains. One of the first was the Proof-of-Existence service [1]. This service creates an ephemeral Bitcoin address and generates a transaction that includes a provided document hash (specifically, it includes an `OP_RETURN` script opcode in the transaction output that includes the marker bytes `DOCPROOF` and the document hash). Similar services exist that can place document hashes on other blockchains, such as [10] for Ethereum and [20] for EOS.

[1]The BoCA software can be found here: https://github.com/Gz75y45ms3kc/boca

ICFNDS 2021, December 15–16, 2021, Dubai, United Arab Emirates

In [6], the authors discuss the advantages of placing evidence on the blockchain, in particular, transparency and immutability. Their proposed system involved placing base64 encoded evidence files on a private blockchain using Hyperledger. The data being encoded would mean posting a considerable amount of data. The use of a private blockchain would alleviate some of the cost that would be associated with using a public blockchain, but this lessens the transparency and security obtained.

The importance of work in this area was highlighted by NIST in [11]. They determined 64 Cloud Forensic Challenges, of which the 24^{th} was "Data chain of custody".

In [23], the authors identified two key requirements for any system to increase the trustworthiness of capturing data from the cloud: anti-tampering and privacy. In their proposed solution, a receiver can request data from a sender and have signed records of their actions and the data be placed on the blockchain by a Provenance Auditor (PA). They used the ChainPoint protocol [4], which groups data via a Merkle tree, reducing the cost of placing data on the blockchain. The PA and Certificate Authority were identified by the authors as points of weakness, i.e., centralised entities that could undermine the integrity of the processes.

In [14], the authors proposed a system where hashes of data are placed on the Ethereum blockchain. They combine this with encryption of the data for confidentiality. However, their solution relies on a trusted central system that stores fragments of the files. The problem with this approach is that while the whole file may not be recoverable, sensitive information can leak from any fragments. The system does not have a comprehensive key management system, requiring private keys to be "sent to the user's designated mailbox".

The use of private blockchains to protect supply chain data was explored in [8]. Their proof-of-concept produces a verifiable audit log of all access to encrypted data (including searching). And while using a private blockchain and other centralized components does improve performance and can ease integration with existing systems, it does come with a trade-off. Controlling access with username/passwords via a web front-end cannot equal the security of the standard (public) blockchain model where all actions are authorised with a private key.

There are numerous decentralised storage services that are build on/incorporate blockchain technology. These include Filecoin/InterPlanetary File System (IPFS) [2], Sia [22], and Storj [19]. These generally use blockchains as a way to provide an incentive mechanism for participants to provide storage and do not offer timestamping/data verification as a feature. Several of them offer high privacy guarantees, which would preclude any transparency or verifiability.

Memo.cash [5] operates in a similar manner as [1] in that it posts data to the blockchain using the `OP_RETURN` script opcode. Where Memo.cash differs is that it uses the Bitcoin Cash (BCH) blockchain instead of the Bitcoin (BTC) blockchain (primarily for the benefit of the former's lower transaction fees). Also, where [1] places document hashes on the blockchain, Memo.cash places data on the blockchain directly. Their aim is to create an entire social network on the blockchain. Users' identities are directly linked to their blockchain addresses and all posts, replies, "likes", and account management actions are done using on-chain transactions.

From reviewing the literature, certain trends become apparent. Numerous proposals have been made that offer additional features beyond simply recording data on a blockchain. However, many of these require the addition of various trusted intermediaries, which dilutes one of the benefits of using blockchains that do not require any explicit trust. Little use is made of the strong authentication implicit in owner of any blockchain address (i.e., the entity with the corresponding private key), and where it is used ([5]) it is done with a very specific purpose. We believe that the potential for blockchain-based data preservation systems is not yet being fully exploited.

2 REQUIREMENTS

As we have already stated, there exists several services that can place document hashes onto a blockchain. Therefore, any novel data preservation system must have new features that provide meaningful improvements. We propose a robust and novel system for preserving data should have the following requirements:

- Integrity: The primary concern is that the data has strong assurances against any modification.
- Authorship and Ownership: It is possible to attribute the data to a specific identity.
- Verifiability: Not only is the data's integrity preserved, but it can be also verified by third-parties.
- Timestamp: The time and date on which the data was released can be preserved and verified to the same confidence as the content.
- Disintermediation: No unnecessary trusted entities are included in the system.
- Decentralised: There is no single point of failure.
- Comprehensive security: The system is presented as a complete solution with no gaps.
- Confidentiality: Where necessary, access to data can be limited to authorised users (and consequently the ability to verify the data is similarly limited).
- Takes advantage of existing components: There should be no redundant systems running in parallel with the blockchain.
- Lightweight: Minimal additional components/infrastructure required.
- Interoperability: The use of, and extension of, standard representations to facilitate integration and development.

While there are existing proposals that satisfy some of the requirements, in our assessment of the state-of-the-art there is none that satisfies all of them.

3 DESIGN

Any data that is placed on a blockchain is preserved with the cryptographic protection of the proof-of-work. This is what allows cryptocurrency transactions with values in millions of Euros to be executed with confidence. The systems mentioned in the Section 1 use this method (in slightly different implementations). What is not generally used to good effect is the significance of the address that is making the transaction. The act of creating a transaction involves signing the transaction details, meaning that it can only

be done by someone who has access to the corresponding private key. This has the potential to offer several novel features.

A Bitcoin or Ethereum address is an identifier that relates to a public-private key pair. The address is the hash of the public key, which means that if only the address is known then it cannot be used for any cryptographic operations. However, once a transaction is made from that address, then the public key is published on the blockchain and is accessible to all. The growth in the use of cryptocurrencies has lead to a sizeable amount of best practices to ensure transactions are completed correctly. These include:

- Protect your own private keys
- Verify the recipient address after copying/scanning the QR code
- Wait for several block confirmations before accepting the transaction
- Verify the transaction in your wallet/on a public block explorer

For additional examples of recommended cryptocurrency security practices, see [21]. While the purposes of these actions are to ensure the correct transfer of funds, part of what the above precautions are doing is verifying of the recipients public key. This can be repeated in the opposite direction. Alice can attempt to send an amount of cryptocurrency to Bob and Bob can attempt to send an amount of cryptocurrency to Alice. If they are both satisfied that the transactions happened correctly (using the established best practices for usage of cryptocurrency), then they have completed a mutual authentication of each other's public keys. This opens up several opportunities for building useful features on top of existing approaches.

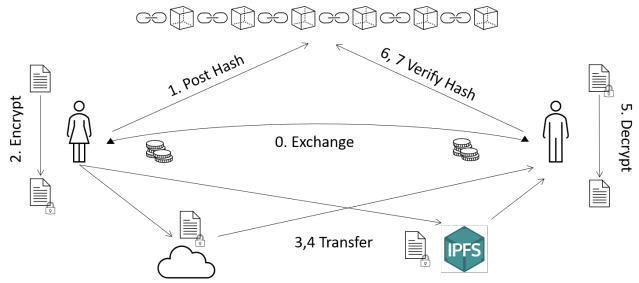

Figure 1: Overview of BoCA.

We propose a novel system for preserving data integrity called BoCA (Blockchain-of-Custody Application). The applications operates as follows (see also illustration in Figure 1). Two users, Alice and Bob, will each generate their own blockchain wallets and addresses. They will each fund their wallets with a small amount of the appropriate cryptocurrency. They will obtain each others' addresses and attempt to send a small amount of cryptocurrency to each other. Once they have verified that the exchanges have occurred, they can use BoCA for the following:

(1) Alice uses BoCA to post a hash of a file on the blockchain.
(2) Alice can optionally encrypt the file with Bob's public key (or with the public key of any users with which she has performed an exchange). The hash of the plaintext file is added to the blockchain (it is better practice to hash/sign plaintext files rather than encrypted files).
(3) Alice makes the file available to Bob using any desired means (not included in BoCA).
(4) Alice can upload the data to the InterPlanetary File System (IPFS) network, rather than needing to rely on a third-party service to transfer the data to Bob.
(5) Bob can use his private key to decrypt any encrypted file received.
(6) On receipt of the file, Bob can use BoCA to find all addresses with which he has performed a mutual exchange, check all those addresses for BoCA posts, and if one has a post of the file hash then confirm the autheniticity (and authorship and timestamp) of the file.
(7) Bob can also provide the poster's blockchain address/blockchain transaction ID as arguments to BoCA to reduce the need for searching/if he has not performed an exchange with Alice.
(8) Alice can include the IPFS CID in the post to the blockchain as well as the hash, which allows Bob (or anyone else) to obtain the data as well as verify its integrity.

BoCA can accept folders as well as files. On posting a folder, an index file will be created with hashes of each individual file as well as some metadata provided by Alice. The hash of this index file is what is posted to the blockchain. Support is also included for the use of keyed HMACs instead of unkeyed hashes. In the case where the disclosure of a hash of the data could leak information, Alice can choose to post a HMAC to the blockchain instead.

We believe that our proposed proof-of-concept, described in Section 4 satisfies these requirements.

4 IMPLEMENTATION

4.1 Blockchains

BoCA supports posting data to the Bitcoin Cash and Ethereum networks, as well as the Ethereum Ropsten testnet. Supporting multiple blockchains gives redundancy in general (if one of the two blockchains suffers a catastrophic failure, BoCA could continue to operate on the other blockchain) and for users in need of additional integrity assurance (a file hash could be posted to both blockchains). The different blockchains offer different benefits: Ethereum is generally considered the larger blockchain by most metrics (see [3]), and so its proof-of-work carries greater "weight", whereas Bitcoin Cash has very low transaction fees which offers a cheaper alternative. The fact that BoCA operates on two blockchains indicates that it could be adapted to further platforms if necessary. Supporting multiple blockchains also opens the opportunity to have protection assured by multiple blockchain consensus mechanisms, as will be the case when Ethereum eventually migrates from Proof-of-Work to Proof-of-Stake.

4.2 Setup

BoCA can generate public-private keys (and the corresponding addresses) for the supported blockchains. Most cryptocurrency wallets use a seed to derive many key-pairs, each of which is intended for brief/individual use. However, as BoCA is using the key-pairs for a persistent identifier, this approach is not suitable. The BoCA

Table 1: Structure of BoCA encrypted file.

Content	Length (bytes)	Purpose
`boca`	4	Identifies the file type is a BoCA encrypted file
(Integer)	1	Identifies the BoCA version (currently 1)
(Integer)	2 (big-endian)	Specifies the length of the encrypted key
(encrypted)	variable	The encrypted key
(encrypted)	variable	The encrypted file

key generator creates one single public-private key-pair for each blockchain, which is encrypted with a user-supplied password (using PBKDF2 [15] with 1 million iterations to derive the key from the password and AES-256-GCM [7] for the encryption). Any user is, of course, free to generate many key-pairs (and save them in different files) which they can use for separate purposes.

4.3 Message Format

On the Bitcoin Cash blockchain, BoCA extends the Memo.cash protocol [5]. Each post starts with a four character identifying string:

- `bocf` - Post contains a hash of a single file.
- `bocl` - Post contains a hash of a list of files in a directory.

The SHA256 hash of the file/list then follows in hex format. The binary values of the hash digest could have been included, but the decision was made to present it as hex. The benefit is that when viewing the transaction details on a block explorer[2] then the hash can be directly viewed. This is at the cost of increased transaction size, which means slightly higher transaction fees. The IPFS CID can optionally be placed after the hash digest.

Ethereum has a `data` field in each transaction. BoCA uses this to include the initial identifying string, the hash digest, and optionally the IPFS CID similar to the format in the BCH blockchain.

4.4 Encryption

When Alice wishes to encrypt the file being preserved with BoCA, she and Bob perform a mutual exchange, which makes their respective public keys available on the blockchain (as well as allowing them to verify each other's addresses). Both Bitcoin Cash and Ethereum use the secp256k1 elliptic curve [17]. Alice generates a random AES key, which is encrypted using the Bob's secp256k1 public key (using the `ecies` python library[3]). This AES key is then used to encrypt the file using AES-256-GCM [7]. The encrypted file is structured as indicated in Table 1. Once encrypted, Alice can make the file available to Bob using BoCA's integration with IPFS, or via any other preferred mechanism.

If the content is a folder, then it is collected into a zip archive before being encrypted.

4.5 Code Overview

The majority of the code of BoCA is within a package of the same name. This has three separate files, `blockchain.py`, `crypto.py`, and `ipfs.py`, which contain various necessary functions as illustrated in Table 2. The functionality of BoCA is provided by seven python files described in the same table.

4.6 Examples

In Table 3, we see an example of a user posting a hash of a file to the BCH blockchain. The file in question just contains the string "abcd" (no new-line) and the transaction can be confirmed on any blockexplorer[4]. Table 4 shows the another user verifying the post using the transaction ID and then the poster's address.

In Table 5, we see an example of two users performing a mutual exchange. After this, the first user (Alice) is able to encrypt a file and upload it to IPFS (Table 6). This file is just the string "123" (no new-line). Since Alice added the `--addcid` option, Bob can simply run the `boca_verify_ipfs.py` script without needing to specify transaction or address details. All mutual exchanges will be determined, Bob can choose which one he is interested in, any "`addcid`" posts will have their IPFS content downloaded, decrypted and verified (Table 7).

5 EVALUATION

5.1 Integrity

The primary requirement of this proof-of-concept is the integrity of the data. As explained earlier, any data preservation system that is proposed with a novel blockchain may be theoretically sound but has a very real practical downside. The strength of any consensus mechanism depends on the number and variety of participants and getting any new blockchain off the ground can be a struggle. BoCA is built on two well-established blockchains and has no such boot-strapping problem.

BoCA has a clear advantage against any data preservation system based on an blockchain model that has as yet been unrealised, but a valid question is to ask just how much integrity is offered. A useful comparison can be made by looking at the practices of the major cryptocurrency exchanges. They regularly participate in transactions that can be valued in the millions of euros, and need to determine some threshold number of block confirmations that are sufficient to consider the transaction immutable. On Kraken, 20 confirmations are necessary for Ethereum and 15 for Bitcoin Cash [13]. On Huobi, 12 confirmations are necessary for Ethereum [12]. On FTX, 10 confirmations are necessary for Ethereum and 2 for Bitcoin Cash [9]. BoCA does not similarly implement any of the above precautions, it just indicates to the user how many confirmations there have been. This is because BoCA may be used by individuals with varying degrees of risk tolerance. However, it would be a simple matter to include a user specified risk level and flag any transaction that has not yet reached the most conservative of the above conditions as "Pending".

[2] Such as https://explorer.electroncash.de/ or https://etherscan.io/
[3] https://pypi.org/project/eciespy/
[4] Such as: https://blockchair.com/bitcoin-cash/transaction/e38770b00a1445f3c0b971a3bd39fa15328efd87d1256b783fe94db3d43d0c0f

Table 2: BoCA Python files

Package file name	Contents/Purpose
`boca/blockchain.py`	Functions used to obtain and parse data from the blockchain as well as posting data to the blockchain.
`boca/crypto.py`	Functions used to manage the users private keys and encrypt/decrypt data using the relevant public/private keys.
`boca/ipfs.py`	Functions to upload/download data to/from IPFS via a locally running daemon. Also includes functions for unzipping zip files, hash calculation and hash verification.
General use file name	**Contents/Purpose**
`boca-keygen.py`	Generate blockchain keys/addresses and save in a password protected file.
`boca-spend.py`	Send an amount of cryptocurrency to a specified address, necessary for the mutual exchange.
`boca-post.py`	Calculate a hash of a file/list of files and post it to a blockchain. Optionally encrypts.
`boca-post-ipfs.py`	Same as `boca-post.py` but content is also uploaded to IPFS.
`boca-verify.py`	Check if hash of content matches hash from the blockchain. Optionally decrypts.
`boca-verify-ipfs.py`	Same as `boca-verify.py` but content is also obtained from IPFS.
`boca-scan.py`	Continually scan the blockchain for any posts from addresses with mutual exchanges, obtain IPFS CID from the blockchain posts, download and verify content.

Table 3: Posting a file hash using BoCA

```
(alice)$ python boca_post.py --path text.txt
Enter password to access private key(s):
BCH address: bitcoincash:qrmcrjgmfu90s62x5ytwmzu8ktdnj2sgl5wd3wqm57
Attempting to post to BCH blockchain
Transaction id: e38770b00a1445f3c0b971a3bd39fa15328efd87d1256b783fe94db3d43d0c0f
```

Table 4: Verifying a file using BoCA

```
(bob)$ python boca_verify.py --path text.txt --ID
e38770b00a1445f3c0b971a3bd39fa15328efd87d1256b783fe94db3d43d0c0f
Transaction provided, attempting to obtain hash from blockchain
Hash of local file text.txt matches hash in unconfirmed transaction on BCH blockchain by
bitcoincash:qrmcrjgmfu90s62x5ytwmzu8ktdnj2sgl5wd3wqm57
(bob)$
(bob)$ python boca_verify.py --path text.txt --address
bitcoincash:qrmcrjgmfu90s62x5ytwmzu8ktdnj2sgl5wd3wqm57
Searching for transactions involving: bitcoincash:qrmcrjgmfu90s62x5ytwmzu8ktdnj2sgl5wd3wqm57
Hash of local file text.txt matches hash placed on BCH blockchain on 2021-08-25 17:40:58 UTC by
bitcoincash:qrmcrjgmfu90s62x5ytwmzu8ktdnj2sgl5wd3wqm57 with 2 confirmation(s) with TXID
e38770b00a1445f3c0b971a3bd39fa15328efd87d1256b783fe94db3d43d0c0f
```

5.2 Authorship and Verifiability

BoCA allows for the identity of the poster of any content to be identified (in terms of their pseudonymous blockchain address). This is verifiable by anyone using a block explorer (the transaction IDs of the example operations have been provided, along with the file content description, to allow inspection and confirmation). We can be assured of the connection to from the author to the address by the need for the private key in creating the transaction (an assurance that does admittedly rely on the private key being kept private). Tying this address to a real world identity has been deliberately left out-of-scope. If needed, any entity is free to publish their address on their website/social media as appropriate, or share it privately. If the identity of any data posted to the blockchain was questioned, they could simply sign any offered challenge with the same private key (or equally post the challenge to the blockchain).

5.3 Confidentiality

All encryption algorithms chosen for this proof-of-concept reflect the currently accepted best practices. BoCA can offer strong assurances of confidentiality of the data entrusted to the system with two caveats. Firstly, the data is encrypted with a private key. If the private key is disclosed, then no confidentiality can be provided. We

Table 5: Performing a mutual exchange using BoCA

```
(alice):$ python boca_spend.py --address bitcoincash:qp9vrpdcswjgrr4qm328sc2gdz7qmdnevvk546s76p
--amount 0.0005
Enter password to access private key(s):
BCH address: bitcoincash:qrmcrjgmfu90s62x5ytwmzu8ktdnj2sgl5wd3wqm57
BCH balance: 0.00175571 BCH or 175571 satoshi
Transaction ID: 5cf7aff16130352cc822e2c9acf544864ef0b20b768d8537eb222054f2d1995d
```

```
(bob):$ python boca_spend.py --address bitcoincash:qrmcrjgmfu90s62x5ytwmzu8ktdnj2sgl5wd3wqm57
--amount 0.00025
Enter password to access private key(s):
BCH address: bitcoincash:qp9vrpdcswjgrr4qm328sc2gdz7qmdnevvk546s76p
BCH balance: 0.00059774 BCH or 59774 satoshi
Transaction ID: 97916d357b4d8800e9d7374f7abe0a80ce43816493a777717c5ed385f7a9b258
```

Table 6: Posting an encrypted file using BoCA

```
(alice):$ python boca_post_ipfs.py --addcid --path text123.txt --enc
Enter password to access private key(s):
BCH address: bitcoincash:qrmcrjgmfu90s62x5ytwmzu8ktdnj2sgl5wd3wqm57
Warning! This option makes your data publicly available. Please use with care.
Attempting to post to BCH blockchain
Searching for mutual exchanges ...
Only a single address that has a mutual transfer found. Using:
bitcoincash:qp9vrpdcswjgrr4qm328sc2gdz7qmdnevvk546s76p
File uploaded to ipfs: QmUUW9q7RDqKUgX7ULpKfod54pGLkwTcFcE91Yx4N1saHV
Transaction id: 2a68120e7a6cac9c5c51d6737d7f91fd2a22bffe070d90a74b26e08433355b48
```

Table 7: Retriving and verifying an encrypted file using BoCA

```
(bob)$ python boca_verify_ipfs.py --getcid --dec
Enter password to access private key(s):
BCH address: bitcoincash:qp9vrpdcswjgrr4qm328sc2gdz7qmdnevvk546s76p
Attempting to verify post to BCH blockchain
Searching for any address with mutual exchange ...
Mutual exchanges occurred with the following addresses:
1) bitcoincash:qpwt6xuc00ntwyx8ypje7xnf2wpznhmusq4jpf2x7e
2) bitcoincash:qrmcrjgmfu90s62x5ytwmzu8ktdnj2sgl5wd3wqm57
Please enter your choice of above [1-2]: 2
You have selected bitcoincash:qrmcrjgmfu90s62x5ytwmzu8ktdnj2sgl5wd3wqm57
Searching for transactions involving:
bitcoincash:qrmcrjgmfu90s62x5ytwmzu8ktdnj2sgl5wd3wqm57
Downloaded file from IPFS: QmUUW9q7RDqKUgX7ULpKfod54pGLkwTcFcE91Yx4N1saHV
File has a BoCA header, version number: 1
File successfully decrypted and verified (with symmetric key).
Hash of local file qrmcrjgmfu90s62x5ytwmzu8ktdnj2sgl5wd3wqm57
/QmUUW9q7RDqKUgX7ULpKfod54pGLkwTcFcE91Yx4N1saHV.dec matches hash placed on
BCH blockchain on 2021-08-26 16:02:20 UTC by
bitcoincash:qrmcrjgmfu90s62x5ytwmzu8ktdnj2sgl5wd3wqm57 with 6 confirmation(s)
```

have used a key file protected with a password (with a suitable key derivation function), but in theory the system could incorporate the use of hardware wallets for greater secrecy.

Secondly, it is generally considered inadvisable to use the same keys for multiple purposes. In the case of BoCA, we used the same private key for decryption as is used for signing transactions. If the user was manipulated to perform a particular private key operation in one context, there is a risk that it could be used in the other. Since the sender decides the parameters when signing, the more

plausible attack would be to induce a decryption in order to forge a signing transaction.

There are two ways to mitigate this. The BoCA wallet should only hold as much funds as is needed for the expected transactions (the benefit of using Bitcoin Cash is low transactions, so the amount of funds that are ever at risk can be kept small). The BoCA verification and decryption tool provides only very sparse information on decryption. Either the content is decrypted and saved to disk or it is invalid, in which case an error message is displayed but nothing is logged. There is no obvious mechanism for an attacker to get any usable information from the output.

5.4 Cost

The example BoCA operation shown in Table 3 costs 0.00000275 BCH, which at current prices amounts to 0.001562 Euro[5]. The example shown in Table 6 (which involved posting the IPFS CID as well as the file hash) costs 0.00000471 BCH, which amounts to 0.002599 Euro. An Ethereum transaction using BoCA costs 0.00167 ETH, which at current prices amounts to 5.41 Euro[6]. Given the costs of posting, it would highly cost-efficient to use BoCA on the Bitcoin Cash blockchain where integrity assurances were required (even with high frequency of activity). Ethereum has seen considerable volatility, both in terms of the price of Ether and in transaction fees. There is a roadmap to address this in future updates [18], but for the time being using Ethereum directly could be too costly for BoCA. Where the assurance of using the Ethereum blockchain was necessary, it would be feasible to perform occasional transactions on Ethereum (containing a hash of a collection of content) with the majority of individual transactions performed on the BCH blockchain.

It would be possible to make small changes to reduce these costs. The hashes are posted as encoded hex when they could be the raw binary instead. However, this would reduce the verifiability and transparency offered by BoCA. The way it has been implemented, the encoded hex are displayed in the transaction details of blockexplorers[7]. It is important to have a way to use a third-party system to obtain the hashes from the blockchain and hence verify the integrity of any document, rather than solely relying on the BoCA software to simply say it is valid.

The second change that could be argued is that since an IPFS CID is a type of hash of a document, it should not be necessary to include both the SHA256 and the IPFS CID in a transaction. However, the IPFS CID serves a different purpose (it is used to obtain the content from the IPFS network) than the SHA256 hash. Also, relying solely on the IPFS CID for integrity would be less than ideal as the ability to create IPFS CIDs is much less commonly available then that of SHA256.

5.5 Efficiency

BoCA is simple in its design. There are no extraneous entities, systems, bottle-necks, nor single points of failure. The sender and receiver need only the BoCA software and the relevant cryptocurrency to safely preserve and exchange data. A third-party verifier would not even need the BoCA software, and could verify any content with just access to a block explorer and any hashing tool. The simplicity and minimalism of the design has benefits for both security and ease of use. Fewer components means fewer things that can go wrong and fewer trusted entities means lower risks of compromise. The re-purposing of the wallet private keys for encryption means a parallel public key infrastructure does not need to be created and managed. This simplicity passes on to the implementation, which allows for users to be active and placing content on the blockchain with minimal overhead. The decision to extend the existing memo.cash protocol and use the existing "data" field in Ethereum transactions improves the interoperability of the solution.

The BoCA software uses third-party APIs for obtaining information from, and posting transactions to, the blockchains. And while this is common practice, it is a reliance on centralised entities and unnecessary intermediaries. However, this is a proof-of-concept. If necessary, a BoCA client could be written to communicate with the relevant blockchain networks directly.

6 FUTURE WORK

While the BoCA application has many benefits, there are a number of potential areas of future work.

- Currently, files are encrypted for a single recipient. It would be possible to instead encrypt a file for multiple recipients using their public keys (obtained in the same way through mutual exchanges).
- A local IPFS client does require port forwarding or another means of traversing NAT. Where this is not practical, it would be possible to add support for a third-party IPFS service[8].
- BoCA manages identities in terms of blockchain addresses/public keys, but more could be done to manage the relationship between real world identities and addresses. One area worth exploring in particular would be how to revoke an address should the private key be disclosed.

The immutable nature of blockchains has the potential to provide exceptionally strong assurances of integrity and authentication. We believe BoCA showcases some of this potential. And while private blockchains have their specific use cases, their reliability and verifiability can be best exploited by applications that BoCA that are built in a way to take advantage of the existing explorers, libraries, and platforms.

7 CONCLUSION

We have presented a novel proof-of-concept that is flexible enough to preserve any data with a high degree of confidence. There are numerous potential applications for this system:

- Preservation of digital evidence for legal proceedings
- Protection of environmental data to assure transparency and reduce fraud
- Documentation of coursework by students to assure academic integrity

[5] https://currencio.co/btc/eur/0.00000275/ Note that prices may have fluctuated wildly since time of writing.
[6] https://currencio.co/eth/eur/0.00167/
[7] In the case of BCH, only those that support memo.cash.
[8] Such as https://infura.io/

- Providing both confidentiality and integrity of patient medical records

By satisfying the requirements of disintermediation, decentralisation, taking advantage of existing components, and being lightweight, the resulting application is simple enough to be easily integrated in any domain/environment. At the same time, the wide array of features offered have the potential to greatly enhance any system.

ACKNOWLEDGMENTS

This work was sponsored, in part, by the ERDF Greater Manchester Cyber Foundry project.

REFERENCES

[1] Christopher Adams. 2017. Proof of Existence. Retrieved November 17, 2021 from https://proofofexistence.com/ [online] https://proofofexistence.com/.

[2] Juan Benet. 2017. Filecoin: A Decentralized Storage Network. Retrieved November 17, 2021 from https://filecoin.io/filecoin.pdf [online] https://filecoin.io/filecoin.pdf.

[3] BitInfoCharts. 2021. Cryptocurrency Statistics. Retrieved November 17, 2021 from https://bitinfocharts.com/ [online] https://bitinfocharts.com/.

[4] Chainpoint. 2020. A scalable protocol for anchoring data in the blockchain and generating blockchain receipts. Retrieved November 17, 2021 from https://www.chainpoint.com/ [online] https://www.chainpoint.com/.

[5] Jason Chavannes. 2018. Introducing Memo. Retrieved November 17, 2021 from https://memo.cash/blog/introducing-memo [online] https://memo.cash/blog/introducing-memo.

[6] M. Chopade, S. Khan, U. Shaikh, and R. Pawar. 2019. Digital Forensics: Maintaining Chain of Custody Using Blockchain. In *2019 Third International conference on I-SMAC (IoT in Social, Mobile, Analytics and Cloud) (I-SMAC)*. 744–747.

[7] Morris J Dworkin. 2007. *Sp 800-38d. recommendation for block cipher modes of operation: Galois/counter mode (gcm) and gmac.* National Institute of Standards & Technology.

[8] Gregory Epiphaniou, Prashant Pillai, Mirko Bottarelli, Haider Al-Khateeb, Mohammad Hammoudesh, and Carsten Maple. 2020. Electronic Regulation of Data Sharing and Processing Using Smart Ledger Technologies for Supply-Chain Security. *IEEE Transactions on Engineering Management* 67, 4 (2020), 1059–1073. https://doi.org/10.1109/TEM.2020.2965991

[9] FTX. 2021. Blockchain Deposits and Withdrawals. Retrieved November 17, 2021 from https://help.ftx.com/hc/en-us/articles/360034865571-Blockchain-Deposits-and-Withdrawals [online], https://help.ftx.com/hc/en-us/articles/360034865571-Blockchain-Deposits-and-Withdrawals.

[10] Acronis International GmbH. 2021. Acronis Notary. Retrieved November 17, 2021 from https://kb.acronis.com/content/64943 [online] https://kb.acronis.com/content/64943.

[11] Martin Herman et al. 2020. NIST Cloud Computing Forensic Science Challenges. Retrieved November 17, 2021 from https://www.nist.gov/publications/nist-cloud-computing-forensic-science-challenges [online] https://www.nist.gov/publications/nist-cloud-computing-forensic-science-challenges.

[12] Huobi. 2021. Announcement on the Adjustment of BTC and ETH Network Confirmation Times. Retrieved November 17, 2021 from https://www.huobi.com/support/en-us/detail/360000334722 [online], https://www.huobi.com/support/en-us/detail/360000334722.

[13] Kraken. 2021. Cryptocurrency deposit processing times. Retrieved November 17, 2021 from https://support.kraken.com/hc/en-us/articles/203325283-Cryptocurrency-deposit-processing-times [online], https://support.kraken.com/hc/en-us/articles/203325283-Cryptocurrency-deposit-processing-times.

[14] Hongyu Li, Liehuang Zhu, Meng Shen, Feng Gao, Xiaoling Tao, and Sheng Liu. 2018. Blockchain-based data preservation system for medical data. *Journal of medical systems* 42, 8 (2018), 1–13.

[15] Kathleen Moriarty, Burt Kaliski, and Andreas Rusch. 2017. Pkcs# 5: Password-based cryptography specification version 2.1. *Internet Eng. Task Force (IETF)* 8018 (2017), 1–40.

[16] Satoshi Nakamoto. 2008. Bitcoin: A Peer-to-Peer Electronic Cash System. Retrieved November 17, 2021 from https://bitcoin.org/bitcoin.pdf [online] https://bitcoin.org/bitcoin.pdf.

[17] Minghua Qu. 1999. Sec 2: Recommended elliptic curve domain parameters. *Certicom Res., Mississauga, ON, Canada, Tech. Rep. SEC2-Ver-0.6* (1999).

[18] C Schwarz. 2019. Ethereum 2.0: A Complete Guide. *Medium* (2019). Retrieved November 17, 2021 from https://medium.com/chainsafe-systems/ethereum-2-0-a-complete-guide-d46d8ac914ce [online] https://medium.com/chainsafe-systems/ethereum-2-0-a-complete-guide-d46d8ac914ce.

[19] Storj Labs. 2018. Storj: A Decentralized Cloud Storage Network Framework. Retrieved November 17, 2021 from https://www.storj.io/storjv3.pdf [online], https://www.storj.io/storjv3.pdf.

[20] Cybrosys Technologies. 2021. Blocktick: Issue and Verify Records using Blockchain. Retrieved November 17, 2021 from https://www.cybrosys.com/blocktick/ [online] https://www.cybrosys.com/blocktick/.

[21] Kevin Ting. 2017. Cryptocurrency security and safety best practices. Retrieved November 17, 2021 from https://www.bitcoinforbeginners.io/cryptocurrency-guide/cryptocurrency-security/ [online] https://www.bitcoinforbeginners.io/cryptocurrency-guide/cryptocurrency-security/.

[22] David Vorick and Luke Champine. 2014. Sia: Simple Decentralized Storage. Retrieved November 17, 2021 from https://academy.bit2me.com/wp-content/uploads/2021/05/SIA-WHITEPAPER.pdf [online] https://academy.bit2me.com/wp-content/uploads/2021/05/SIA-WHITEPAPER.pdf.

[23] Y. Zhang, S. Wu, B. Jin, and J. Du. 2017. A blockchain-based process provenance for cloud forensics. In *2017 3rd IEEE International Conference on Computer and Communications (ICCC)*. 2470–2473.