TRUSTED TIMESTAMP IN CRYPTOCURRENCY BLOCK CHAIN

Jan Klusáček
Doctoral Degree Programme (2) , FEEC BUT
E-mail: xklusa00@stud.feec.vutbr.cz

Supervised by: Petr Honzík
E-mail: honzikp@feec.vutbr.cz

Abstract: There are many situations when it is needed to prove that certain information existed previous to some point in time without releasing the actual data. In this article selected schemes of trusted time stamping are described. Most of these traditional schemes require one or more Time Stamping Authority. A new time stamping method based on Bitcoin block chain which does not require any central authority is introduced. Implementation of this method is available at address: http://www.klusacek.tk/~honza/bc.nv/bitcoin.php.

Keywords: Bitcoin, Block chain, Trusted time stamp

1 INTRODUCTION

There are many ways of proving that certain information existed in particular time. The simplest ways is to make the information public. But often it is not suitable to release an information in advance and a more complicated process must be used. This process is called trusted time stamping.

1.1 TRUSTED TIMESTAMP

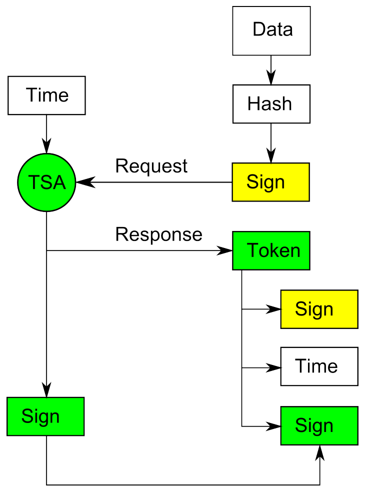
One of the first methods similar to trusted timestamps were anagrams first used by Galileo Galley. From today's point of view this scheme isn't too safe. An anagram reveals lot of information in original message and on the other hand it's often possible to create multiple meaningful messages for a single anagram.

Traditional way to obtain a proof of existence is to use notary services. In this case notary represents trusted third party (TTA). Modern schemes for trusted time stamp (TTS) use a digital signature. One of standards describing TTS is RFC 3161 [1].

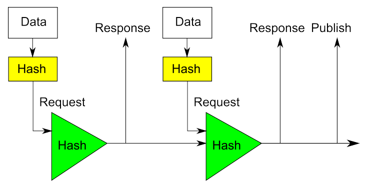
The creation of a trusted timestamp according RFC 3161 is on fig. 1a. Requestor creates a hash or a signature from data and sends it to the Time Stamping Authority (TSA). TSA add a timestamp to the received hash and bind them together with a digital signature using its private key. Hash, timestamp and sign create a token which is send back to requestor. Requestor saves this token for later, when a verification is needed.

When a verification is needed, data and appropriate token from TSA are sent to the verifier. Verifier checks if the token signature belongs to the TSA and compares the hash from the received data and hash in the token. If everything matches the verifier knows that the timestamp is valid. Main weakness of this scheme is that malicious TSA can create token with timestamp in past. There are several possibilities how to mitigate this weakness. One of them is use of transient keys. In this scheme key pairs are generated and assigned to short time intervals, and when the interval expires the private key is destroyed and thus it's not possible to create the token retroactively. Prerequisite for this scheme is that private keys are reliably deleted and cannot be copied.

Another way of increasing TSA trustworthiness is use of linked time stamping. The scheme of this method is on fig. 1b. Tokens created by the linked time stamping contain a hash of the previous token. This way they are entangled to a linear structure. Any later modification of previous timestamp



(a) A creation of trusted timestamp according RFC 3161



(b) A linked time stamping

Figure 1: Trusted timestamp schemes

would change all following tokens. For additional security the top of this structure can be published in some hard to modify media like printed newspapers. Any backdating of issued timestamps is nearly impossible, even for the issuing server.

2 BITCOIN PROTOCOL

Bitcoin is the first of cryptocurrency. It was introduced in 2009 by developer under pseudonym Satoshi Nakamoto. Since then many other cryptocurrencies were created. All cryptocurrencies share the same main principles. They all use cryptography to create secure, distributed and decentralized currency. Communication between clients is established using p2p network. The biggest difference between traditional currency and cryptocurency is the absence of any central authority. This means that there is no easy way of changing properties of existing cryptocurrencies (e.g. total amount of money). It's also impossible to reverse any transaction, even if it is evidently illegal. Basic mechanisms of Bitcoin network will be described below. Other cryptocurrencies are very similar.

2.1 ADDRESS

Bitcoin address is 160bit hash of public key from Elliptic Curve Digital Signature Algorithm (ECDSA) key pair. Each address has its own keypair. Generating a new address is free. All user's addresses with their keys are saved in a file (wallet) used by users's Bitcoin client. This is the only file that the user needs to backup, because private keys are necessary to access user funds. An address is represented in user interface in Base58 encoding which is similar to Base-64 without characters O,0,I and 1.Each address contains 4byte checksum, so mistyped addresses can be detected. If funds are sent to well-formed address for which nobody has the private key, they are lost forever.

2.2 TRANSACTION

Every transaction consists of a transaction id, a list of inputs and a list of outputs. Transaction id is unique identification used to refer to this transaction in the future. Items in the list of the inputs refer to the outputs of the previous transactions. All funds from this transactions will be used. Every input sender has to prove that it is a real recipient of refered transaction. This is most often done by presenting the public key referenced in a previous (input) transaction and signing the simplified version of a new transaction with matching private key[1]. Items in the list of outputs most often [2] contain addresses of recipients and the amount of funds which should be sent to this address. Sum of all outputs has to be smaller or equal to sum of all inputs. The difference between inputs and outputs is used as transaction fee.

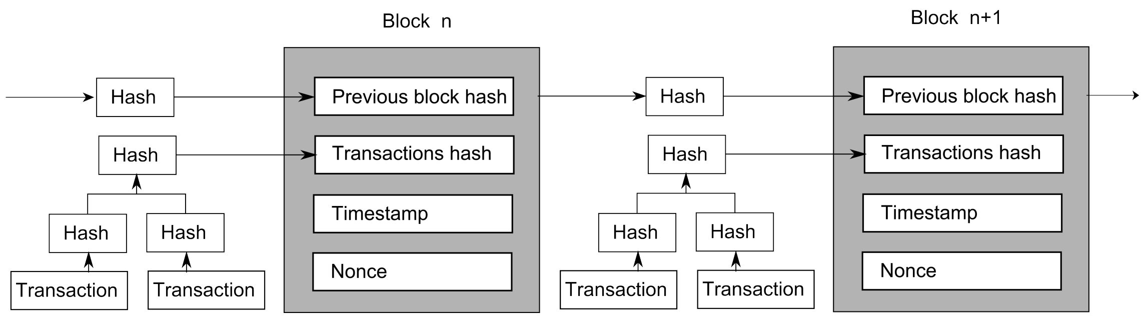


Figure 2: Block chain structure

2.3 Block chain

All transactions are grouped to blocks which form the block chain. Purpose of this chain is similar to the linked time stamping described in 1.1. Simplified structure of these blocks is on Figure 2. The purporse of these blocks is to prevent double-spending and manipulation with accepted transactions and to resolve conflicts. Among other things each block contains: a hash of previous block, transactions received since the previous block, a timestamp and an arbitrary constant (called nonce). To be accepted by the network, the hash of this block has to fulfill certain conditions. First condition is that all transactions have to be valid. Another condition is that a hash of this block has to start with a certain number[3] of leading zeros. The used hash function is not invertible so there is no simple way to satisfy the second condition. The only way to find a block fulfilling the second condition is to try different values of the nonce and check if a resulting hash has required properties. This process is very difficult and requires lots of calculations (on average $13 \cdot 10^{18}$ hashes are needed to find one valid block). When one node finds a valid block, it sends it to others. They check if the block is valid and start searching for a next block. The node which finds the valid block receives a reward for finding the block and all fees generated from transactions included in this block. Sometimes two valid blocks are found at almost same time on different nodes. In this case two mutually exclusive blocks are accepted in different parts of the network, effectively forking whole network to two independent parts. To recover from this situation client software employs a rule that prefers the longer block chain. This means that when a second block is found in one of the forked networks its block chain is longer. This longer block chain is than accepted by the other part of the network. Transactions in the alternative block chain between the fork and accepting the longer block chain are lost and have to be send again. This means that if the receiving side of the transaction needs to be sure that transaction is irreversible it has to wait until it is few blocks deeper in the block chain.

3 TIMESTAMP IN BLOCK CHAIN

From the description of the Bitcoin protocol above it is obvious that it employs its own type of TTS. It has many advantages compared to traditional the TTS schemes. It does not require any single trusted

[1]This step is necessary to prevent any alteration of created transaction

[2]Every output contains a script describing how to gain access to it's content. Most often providing public key and signing simplified transaction with appropriate private key is sufficient, but more complex variants are possible. For example, it is possible to request that withdrawing transaction is signed at lest with 6 of 10 keys. This could be used to create accounts with multiple owners. Each owner has one of the keys and the majority of the owners has to agree to move any funds

[3]Number of required zeros is changing according to network performance to keep the average time between blocks at 10 minutes.

authority. It is easy to access for everyone with access to the internet. It is considered very secure[4]. It would be very useful to use these properties not only to create TTS of Bitcoin transactions, but for any arbitrary data. Unfortunately the Bitcoin protocol doesn't support arbitrary messages in the block chain. It is even not possible to add any additional messages to a transaction. This means that any data to be inserted into the block chain has to be converted to look like a transaction. Most straightforward solution is to replace an output address (hash of recipients public key) of the transaction with a hash of data to be time stamped. There is no way to tell if a hash in transaction is a hash of arbitrary data or a hash of public key. Downside of this approach is that Bitcoin sent in this transaction are lost forever (there is no private key with the appropriate hash). There are other ways of encoding arbitrary data to the transaction. One possibility is to encode data to the script part of output transaction. Other possibility is to use data instead of the random integer used for the ECDS signature of a transaction [2]. Later two methods have advantage that Bitcoins from these transactions could be reclaimed. In the current implementation even the transactions which are already spent are stored forever. However there are plans [3] that future clients will be able to discart old spend transactions. This means that if the Bitcoins are recovered, the timestamp could be lost in the future.

3.1 Demo implementation

The implementation described in this paper is using a hash of time stamped data in place of the recipient address. Demo application is available on page http://www.klusacek.tk/h̃onza/bc.nv/bitcoin.php.

It consists of three main parts. The first part is a patch adding several API calls to the orginal Bitcoin client. These calls allow to acces information from the block chain which are not directly accesible using the standard API. The second part is a collection of scripts processing information from the block chain to a sqlite database. This allows to work with them more effectively. The last part is web interface. It allows to create new timestamp from uploaded file and to insert it into the block chain. It can also search for existing timestamp for uploaded file.

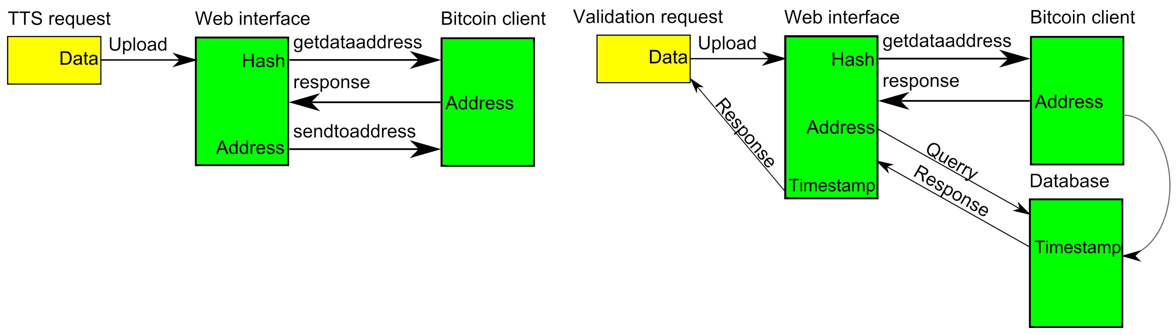


Figure 3: Diagram of TTS creation and validation in described implementation

The process of timestamp creation consists of several steps. The first step is calculating SHA1 hash from an uploaded file. This hash cannot be used directly as a Bitcoin address because it does not have the right lenght and it is missing a required checksum. To form a valid address the API function *getdataaddress* is called. This function was added to the orginal client. Its purpose is to provide access to functions used to derive address from a public key used in the orginal client. This address is then used to send funds by calling API function *sendtoaddress*. Process of a TTS creation and validation is on figure 3

Timestamp is created once the transaction is accepted to block chain. The time between blocks is not constant. It is distributed by exponencial distribution with mean value of 10 minutes. Lengths of time intervals between blocks discoveries gathered from a block chain from block 1 to block 288700 are on figure. 4. This histogram shows that 95% of blocks are found in 1600 s or less.

[4]Most proposed attacks require access to computer performance comparable to performance of the whole network.

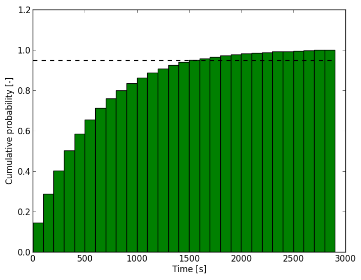


Figure 4: Cumulative probability of block discovery

When a validation of time stamp is required, bitcoin address is generated the same way as timestamp is generated. The transaction with this address in output is then searched in a block chain. Bitcoin client does not allow to search block chain using output as index. Therefore sqlite database containing required information is used to do the search. When the transaction is found it is assigned to its block. Timestamp of this block is then read.

4 CONCLUSION

The main advantage of using a Bitcoin block chain instead of a traditional methods of TTS described in 1.1 are:

Low price In current implementation price of one time stamp is 0.00016 BTC (0.06 USD according exchange rates on 3.3. 2014). Current default settings of Bitcoin client require a fee of 0.0001 BTC for small transactions. Transaction could be set lower but most clients will not accept it to the block chain (it will take long time to be added to block chain).

Trustworthiness Lack of any central authority and proof-of-work system make it very hard to attack.

Simple usage Creation of timestamp and its validation is very easy using the created web interface. Validation of timestamp will be possible even if created interface does not exist anymore (using tools like www.blockchain.info/).

Good accuracy Time accuracy is better than services provided by notary, but worse than accuracy offered by authorites providing commercial TTS.

Demo implementation of described the method is aviable at http://www.klusacek.tk/ honza/bc.nv/bitcoin.php. Timestamp creation is limited to one every hour (to limit costs). Validation is not limited. Digital version of this paper with its time stamp is aviable at http://www.klusacek.tk/˜honza/bc.nv/dv.php

Acknowledgement: This work was supported by grant „Research of Modern Methods and Approaches in Automation“ from the Internal Grant Agency of Brno University of Technology (grant No. FEKT-S-11-6).

REFERENCES

[1] C. Adams, P. Cain, D. Pinkas, and R. Zuccherato. Internet X.509 Public Key Infrastructure Time-Stamp Protocol (TSP). RFC 3161 (Proposed Standard), Aug. 2001. Updated by RFC 5816.

[2] J. Clark and A. Essex. Commitcoin: Carbon dating commitments with bitcoin. *IACR Cryptology ePrint Archive*, 2011:677, 2011.

[3] S. Nakamoto. Bitcoin: A peer-to-peer electronic cash system. May 2009.