C-chain: a system for managing public and private ledgers, an alternative to blockchain

Prof. Rudolf Bayer, TU München Sept. 2017

Summary

Blockchain experiences a hype to manage public ledgers like for the crypto currency bitcoin. But blockchain has a number of very serious drawbacks like

- no scalability,
- bad public image,
- terrible ecological footprint,
- no final settlement,
- high cost of transactions,
- splits of the community.

For details see [4]. Therefore, the blockchain technology is highly questionable.

This paper presents the **C-chain** system as an alternative to blockchain. C-chain integrates cryptographic techniques and database techniques in a novel way in order to manage contracts via public ledgers in a cryptographically certified, secure, immutable and durable way. C-chain avoids the pitfalls of blockchains and has additional substantial advantages.

The techniques integrated in C-chain from cryptography are

1. public private key pairs for encryption and decryption
2. cryptographic signatures.

The techniques from databases are

1. ACID transactions
2. Serialization
3. Integrity constraints

C-chain guarantees security and durability both for the content of contracts as well as for the bookkeeping of contracts in chains of transactions.

Basic Concepts

Laws of public cryptpgraphy

key pairs [π, σ] for Owner U denoted by: **[π_U , σ_U]**

A key pair **[π ,σ]** consists of the public key π and of the private key σ**.**

π is public and can be stored in a public key database **UDB** (**U**ser **D**ata **B**ase), optionally together with additional information. Everybody may see π and query and read UDB. The owner U of π_U may remain anonymous or reveal, who U is as a person or as an organization (authentication) or even as a machine like a socalled IoT device in the Internet of Things (IoT).

σ must be kept secret by the owner and should be copied into a safe place against loss.

Encryption and decryption

Encryption of public documents

Let **d** be an arbitrary document, e.g. the text of a contract or a video.

d is encrypted and signed using the private key σ to compute **σ(d)**. The encrypted version **σ(d)** of d can be decrypted again by using the corresponding public key π and computing $\pi(\sigma(d))$. This reflects the first basic mathematical law of public cryptography: **$\pi(\sigma(d)) = d$**

This law will be used later for encrypting and digitally signing a document. For efficiency, C-chain uses a slightly different version.

Encryption of secret documents

The second basic mathematical law of public cryptography is the reverse of the first: **$\sigma(\pi(d)) = d$** This law is used for encrypting a document in order to hide its content.

If d has been encrypted with the public key π of O resulting in **π(d)**, it can only be decrypted and read by the owner of σ**.** Therefore, if **π(d)** is somehow obtained by anybody, e.g. a user W or even a hacker H by intercepting a message containing **π(d)**, this is completely useless, since he cannot decrypt it.

Both laws together are the basic laws of public cryptography:

$$\pi(\,\sigma(d)\,) \;=\; d \;=\; \sigma(\,\pi(d)\,)$$

Digital Signature

A document d together with the encrypted version **σ(d)** denoted as **[d, σ(d)]** are a mathematical proof that d has not been changed, if the following property holds: **$\pi(\,\sigma(d)\,) = d$**

Since σ is secret, only the owner O of σ can produce **σ(d)** with the property that **$\pi(\sigma(d)) = d$**. Therefore we use σ(d) also as the digital signature of O, proving mathematically, that O has signed the document d and thereby certifies the correctness of d.

Therefore, anybody who knows the public key **π,** the document d and **σ(d)** can verify that d has not been changed (immutability of d).

Note that **σ(d)** has three entirely different aspects:

1. **Encryption**: **σ(d)** is encrypted
2. **Immutability**: if **$\pi(\sigma(d)) = d$** then d has not been changed
3. **σ(d)** has been **digitally signed** by the owner of σ.

Such a digital signature has the following fundamental properties:

1. a digital signature can never be denied by the owner O of σ
2. a digital signature can never be revoked by O
3. a digital signature cannot be forged

Summarizing the basic laws of public cryptography:

$$\sigma(\,\pi(d)) \;=\; d \;=\; \pi(\,\sigma(d))$$

σ(d)   is used in C-chain as a (public) digital signature

π(d)   is used in C-chain for communicating a document secretly

Digital signatures with a hash function

A hash function h is a one-way mathematical function. **h(d)** can be computed easily, but it can be made arbitrarily difficult to compute its inverse **h^{-1}(d).** In C-chain hash functions are chosen in such a way, that collisions are extremely unlikely.

The digital Identity *cryptID*

A digital identity **cryptID** is the pair **[π, σ(h(π))].** For a particular user U his cryptID is denoted by **[π_U, σ_U(h(π_U))]**

The cryptID of U can easily be generated by a device of U and checked by another user V. Therefore, we use the cryptID for opening a communication channel with an arbitrary partner, e.g. with another person, a WEB-service or an arbitrary computer server.

Login to Computers

Now U may use his cryptID and an automaticallly generated and signed PDW to log into computers, servers or WEB-services as follows:

1. The cryptID **[π_U, σ_U(h(π_U))]** is used as the **login name**
2. U uses a signed random string r as the **PWD [r, σ_U(r)]**
3. U now logs in with the pairs **[[π_U, σ_U(h(π_U))] , [r, σ_U(r)]]**
4. This can even be simplified to log in with **[π_U, [r, σ_U(r)]]**

Since r is signed by U, nobody except U could have produced the PWD **[r, σ_U(r)].** Therefore, the server or WEB-service has proof, that this PWD is from U. This simplifies the conventional login process dramatically and at the same time makes it much more secure, since U does not have to remember or to type anything, therefore a key logging malware is no danger.

Login with single use Passwords

Since a signed PWD can be generated in less than a millisecond, a new PWD can be used for every login. This results in **passwords, which are used only once** and would be much more secure than conventional login techniques today. Even if a PWD should be stolen by a *man in the middle attack*, somebody else might be able to log into a system, but for further actions the hacker would need the private key, which he cannot obtain.

The communication between a User U and a server is, of course, secured by https. In addition, a PWD can be made to be valid only for a short time, leaving almost no chance for a man in the middle attack .

Certificates

Foreign certificates

A **foreign certificate** is a signature that two entities A and B belong together. The entities A and B could be a person's legal name or email address and the person's public key. So $\sigma_U([\mathbf{A, B}])$ is a certificate issued by user U himself. By his digital signature U certifies publicly that A and B belong together. But such a certificate is nothing but a claim signed by U, and often such claims are wrong. Foreign certificates are the standard use of certificates.

Self certificate $\sigma_V([\mathbf{V}, \pi_v])$ By this V certifies himself with his own signature σ_V that V and π_v belong together. We used the selfcertificate $\sigma_U(\mathbf{h}(\pi_U))$ in the cryptID of U to prove that π_U ist the public key, that belongs to σ_U. This is much more than just a claim!

Biometric digital certificate for authentication

Authentication is the certification that certain claims are true, for example that a picture was painted by Picasso. For works of art authentication is certified by an expertise and certificates are guaranteed by art experts. But such an expertise is not absolutely certain, it is only a claim! For digital objects we assert authentication by digital signatures and also call them certificates.

In the C-chain system we use the **cryptID** combined with **biometric authentication** to certify that a person is the owner of a public key. For this, U records a video in which he reads the hash $\mathbf{h}(\pi_U)$ of his own public key. This video is then signed and published in the UDB together with π_U **and** $\mathbf{h}(\pi_U)$**,** everybody may see and check it. UDB may contain additional optional information about U.

Users of the C-chain system

We distinguish between the following different types of users:

Normal users denoted by **U, V, W**. They will try to cheat if it is to their advantage and the danger of discovery is low, e.g. to double spend money. But C-chain will prevent that they are successful in cheating.

Trusted Users denoted by **B, C, D**. They are honest and try to follow accepted business rules. They are typically honest, because it is in the best interest of their business, to have a good reputation and to be trusted. If e.g. a pizza service does not deliver the pizzas exactly as ordered, a customer will not order again.

Hackers denoted by H: We will see that hackers have no chance to attack a C-chain System.

Transactions

Business processes in the simplest case consist of a sequence of several isolated and closed transactions, e.g. if U buys a product P from a vendor V and pays via his bank B, this business process consists of the following transactions:

1. U orders P from V
2. V acknowledges the sale
3. V sends P to U
4. V sends the bill to U
5. U orders his bank B to pay
6. B makes the payment to V

All transactions in a business process are combined by C-chain into a strict sequence of transactions and booked as a **transaction chain TC**.

Format of a transaction

A user U formulates a transaction T containing data d and signs it as follows:

> **[d , σ_U (h(d))]** containing d as open data and the signature of the hash of d to assert that d has not been modified. **U is responsible that the content d is correct** and follows the rules of the business involved.

Messages

Public messages M

U sends a message M to V containing the transaction T in the format:

$$\mathbf{M = [\ \pi_U, \pi_V, [\ d\ , \sigma_U (h(d))\ ]\ ]}$$

Everybody who sees this message can check, whether d has been modified or not. This property is called **immutability**.

Secret messages SM

U sends a secret message SM to V with the transaction T in the format:

$$\mathbf{SM = [\ \pi_U, \pi_V, [\pi_V (d),\ \sigma_U (h(d))\ ]\ ]}$$

Hiding the sender of a message HM

This is easily done by also encrypting the sender U in the format

$$\mathbf{HM = [\pi_V(\pi_U), \pi_V, [\pi_V (d),\ \sigma_U (h(d))\ ]\ ]}$$

For efficiency reasons, C-chain uses a different method for encryption consisting of a combination of asymmetric and symmetric encryption.

Rules for transactions

Most transactions must obey certain rules, but it is essential to clearly distinguish between rules for transactions and rules for the bookkeeping of transaction chains. For the above sales process obviously product number, price, account number of V etc. must be correct. The author U of a transaction is responsible that it is formulated correctly. Rules are enforced by U, V and the SW generating transactions. ***Often such rules can be conveniently formulated as integrity constraints in databases, which are then enforced automatically by the DBMS.***

Management of transaction chains

Transaction chains TC are stored in databases TDB (**T**ransaction **D**ata **B**ase). The main task of a TDBMS (**T**ransaction **D**ata **B**ase **M**anagement **S**ystem) is the proper bookkeeping of transaction chains. Independent of the properties and the correctness of the individual transactions, the TC as a whole must have the following properties, for which one or several replicated TDBMS are responsible:

1. Only transactions **certified** by the author with his digital signature are acceptable and added to the transaction chain
2. Once a transaction has been booked in the TC it must be **immutable**, i.e. T cannot be changed after it has been booked, not even by its author
3. TC as a whole must be **strict**, i.e. a T once booked may not be removed from TC and its position within TC may not be changed
4. New transactions may only be **appended** to the end of a chain and not inserted in between. TDBMS is responsible for this. In the context of database systems this property is usually called serialization, but in C-chain this classical form of weak serialization is enforced by unbreakable cryptographic certification.
5. TC must be **durable**, i.e. TC may never disappear and it must be accessible at all times, optionally by the general public, by closed groups or only by the partners of a business transaction.
6. The booking of a transaction must immediately be **settled finally** and forever. If for whatever reason a transaction should have been faulty, it cannot be removed from TC, it can only be compensated by another transaction (like in proper ledgers of bookkeeping), e.g. by repaying the money for a faulty product. But this repayment is a new transaction and the original payment transaction is not removed from TC.

Comment:

Serialization is guaranteed by the TDBMS in C-chain, even if two transactions T1 and T2 arrive at TDBMS exactly at the same time. In such a case the transaction manager of TDBMS decides arbitrarily to book T1 and T2 in some order, e.g. (T1 ; T2) or (T2; T1). The booking result ist not deterministic, but it must be correct.

Protocol of TDBMS

The above requirements of transaction chains are guaranteed by TDBMS by the following protocol, if U wants to send a transaction for V:

1. Before TDBMS appends $\mathbf{T = [\, d, \sigma_U(h(d))\,]}$ to the end of the chain, **TDBMS checks** that T is certified by U by computing **h(d)** and compairing it with $\pi_U(\sigma_U(\mathbf{h(d)})$?
2. **Certification of T by the system TDBMS** with ist own private key σ_S: After checking the proper certification by U, TDBMS now also signs T as
   $$\sigma_S\,(T) = [\, d, \sigma_S(\sigma_U(h(d)))\,]$$
3. Now TDBMS **appends** T with serial number n+1 to the end of the chain TC. If the last T in TC is T_n , then the new transaction is appended to TC as follows:
   a. Read T_n from the end of TC
   b. Compute $h(T_n)$
   c. Book T_{n+1} in the signed form $\mathbf{[\, n+1, \sigma_S(h(T_n))\,, \sigma_S(T)\,]}$ i.e. store this T_{n+1} in TDBMS. $\sigma_S(\mathbf{h(T_n)})$ as part of T_{n+1} makes sure, that T_{n+1} was appended to the chain bei TDBMS
4. The transaction chain TC stored also locally by U is synchronized, so that U can check immediately, that his transaction was properly booked
5. The TC stored locally by V is synchronized by a push notification of TDBMS or as soon as V switches on his client device.

Note: To guard against modification of TC even further, the hash of some component of T_{n+1} could be included in T_n (resulting in a forward linking of TC).

Creating a new Chain: of course, a user U must be able to create a new chain TC. This could be done by using a special document d with the content like this: d = „this chain is named ***bank account of U***. It was created by U on <datetime>“. This record could have the special format:

$$\mathbf{[\, 0, 0, \pi_U, \pi_S\,, [\, d, \sigma_S(\sigma_U(h(d)))\,]\,]}$$

Also, U must be able to control, which other users may interact with V. This is an easy exercise. In this case, V is the recipient of T and decides what to do with T, e.g. to act accordingly, to reject it or to reply in some other manner. But before V sees and acts, T must be properly booked, but this booking process has nothing to do with the content of T, which may even be hidden from TDBMS.

Further essential properties of C-chain

1. Digital identities cryptID **[** $\pi,\ \sigma(\mathbf{h}(\pi))$ **]** are stored in a public database UDB and can easily be found and verified by any user V.
2. To increase security, U could expand his pure cryptID optionally by a photo or video for authentication, by his name, email address, his company name, URL of his social media presence, telephone number etc. Of course, all these data should be signed by U for additional security
3. A user V can then find all information published by U about himself by querying the UDB and downloading it immediately onto his own device. Thus, it is sufficient that V checks the cryptID with the public key and/or authenticity of U only once and very conveniently
4. Perfect scalability
5. Very fast processing
6. Immediate final settlement

Visibility and Rules for Transactions

The rules for transactions must be formulated, obeyed and checked by various agents. Every transaction type has its own rules. For this, certain parameters must be available and visible for the agents involved. The visibility of various components of a transaction can be easily controlled and customized by proper encryption for authorized agents.

References

1. https://blockchain.info
2. http://www.coindesk.com/understanding-dao-hack-journalists
3. https://www.cnet.com/news/in-their-words-experts-weigh-in-on-mac-vs-pc-security/
4. R. Bayer, E. Loibl, C. Roth: Pitfalls of Blockchains, http://www.transaction.de/C-chain/Pitfalls
5. R. Bayer: C-chain, http://www.transaction.de/C-chain/C-chain
6. R. Bayer, E. Loibl, C. Roth: C-chain Anwendungen, http://www.transaction.de/C-chain/Applications
7. R. Bayer: Highlights of C-chain, http://www.transaction.de/C-chain/Applications
8. https://bitcoin.org/bitcoin.pdf
9. https://en.wikipedia.org/wiki/Satoshi_Nakamoto
10. http://www.euroforum.de/best-of-blockchain/#
11. https://digiconomist.net/bitcoin-energy-consumption
12. http://www.wiwo.de/finanzen/geldanlage/studie-zum-zahlungsverkehr-bargeld-ist-teurer-als-kartenzahlung/8232850.htm
13. https://www.welt.de/finanzen/article116392426/So-viel-kostet-die-Deutschen-ihr-Bargeld.html
14. http://www.faz.net/aktuell/finanzen/meine-finanzen/geld-ausgeben/dank-eu-voschrift-muenzen-kosten-mehr-als-ihr-wert-13756893.html
15. http://www.handelszeitung.ch/konjunktur/teure-scheine-was-uns-das-bargeld-kostet-793411
16. https://blockchain.info/de/charts
17. https://de.wikipedia.org/wiki/Liste_der_leistungsstärksten_Kernreaktoren#Europa
18. C‘t 2017 Heft 23: Das macht Blockchain, S. 102-106
19. C‘t 2017 Heft 23: Vertrag denkt mit, S. 108-112
20. superMUC: https://www.lrz.de/services/compute/supermuc/systemdescription/
21. https://paymentandbanking.com/die-disruptivste-zahlmethode/