(12) INTERNATIONAL APPLICATION PUBLISHED UNDER THE PATENT COOPERATION TREATY (PCT)

(19) World Intellectual Property Organization
International Bureau

(43) International Publication Date
04 April 2019 (04.04.2019)

WIPO | PCT

(10) International Publication Number
WO 2019/063512 A1

(51) International Patent Classification:
G06F 21/32 (2013.01)

(21) International Application Number:
PCT/EP2018/075874

(22) International Filing Date:
25 September 2018 (25.09.2018)

(25) Filing Language: English

(26) Publication Language: English

(30) Priority Data:
10 2017 217 342.4
28 September 2017 (28.09.2017) DE

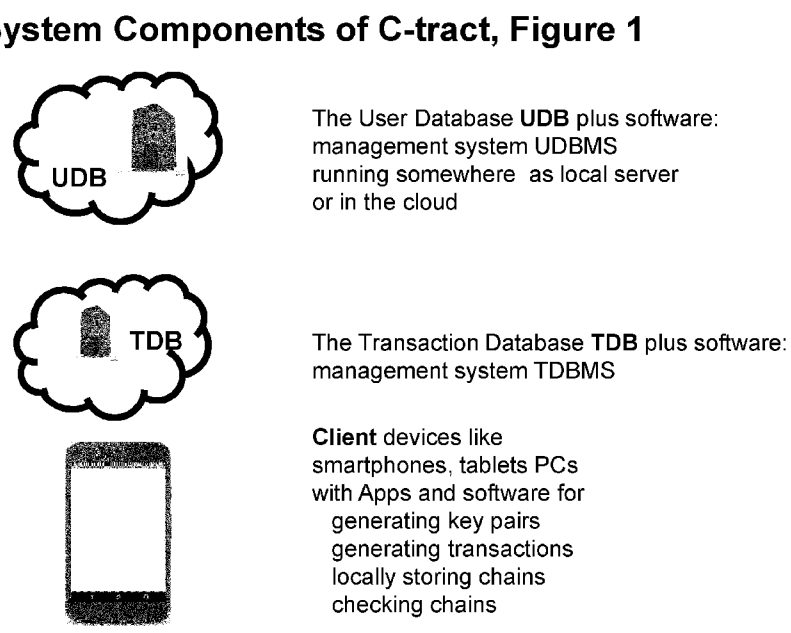
(72) Inventor; and
(71) Applicant: **BAYER, Rudolf** [DE/DE]; Riegseestr. 12, 82194 Gröbenzell (DE).

(74) Agent: **GLAWE DELFS MOLLPARTNERSCHAFT MBB VON PATENT- UND RECHTSANWÄLTEN**; Postfach 13 03 91, 20103 Hamburg (DE).

(81) Designated States *(unless otherwise indicated, for every kind of national protection available)*: AE, AG, AL, AM, AO, AT, AU, AZ, BA, BB, BG, BH, BN, BR, BW, BY, BZ, CA, CH, CL, CN, CO, CR, CU, CZ, DE, DJ, DK, DM, DO, DZ, EC, EE, EG, ES, FI, GB, GD, GE, GH, GM, GT, HN, HR, HU, ID, IL, IN, IR, IS, JO, JP, KE, KG, KH, KN, KP, KR, KW, KZ, LA, LC, LK, LR, LS, LU, LY, MA, MD, ME, MG, MK, MN, MW, MX, MY, MZ, NA, NG, NI, NO, NZ, OM, PA, PE, PG, PH, PL, PT, QA, RO, RS, RU, RW, SA, SC, SD, SE, SG, SK, SL, SM, ST, SV, SY, TH, TJ, TM, TN, TR, TT, TZ, UA, UG, US, UZ, VC, VN, ZA, ZM, ZW.

(84) Designated States *(unless otherwise indicated, for every kind of regional protection available)*: ARIPO (BW, GH, GM, KE, LR, LS, MW, MZ, NA, RW, SD, SL, ST, SZ, TZ,

(54) Title: A METHOD FOR GENERATING A DIGITAL IDENTITY, A DIGITAL IDENTITY, A METHOD FOR CREATING AN ELECTRONIC TRANSACTION DOCUMENT AND AN ELECTRONIC TRANSACTION DOCUMENT



(57) Abstract: A method for generating a digital identity (cryptID) of a user (U) for user authentication in electronic transactions, the user (U) being in possession of a cryptographic key pair (συ, πυ) comprising a public key (πυ) and a private key (συ), the method comprising the following steps: computing a hash function of the public key (πυ) thus generating a public key hash value (h(πυ)); digitally signing the public key hash value (h(πυ)) with the private key (συ) thus generating a signed hash value (συ(h(πυ)); establishing the digital user identity (cryptID) to be the pair consisting of the public key (πυ) and the signed hash value (συ(h(πυ)). The digital identity (cryptID) of a user (U) for user authentication in electronic transactions is comprised of the public key (πυ) of the user (U) and a public key hash value (h(πυ)) digitally signed (συ(h(πυ)) with the private key (συ) of the user (U). An electronic transaction

WO 2019/063512 A1

[Continued on next page]

UG, ZM, ZW), Eurasian (AM, AZ, BY, KG, KZ, RU, TJ, TM), European (AL, AT, BE, BG, CH, CY, CZ, DE, DK, EE, ES, FI, FR, GB, GR, HR, HU, IE, IS, IT, LT, LU, LV, MC, MK, MT, NL, NO, PL, PT, RO, RS, SE, SI, SK, SM, TR), OAPI (BF, BJ, CF, CG, CI, CM, GA, GN, GQ, GW, KM, ML, MR, NE, SN, TD, TG).

Declarations under Rule 4.17:

— *as to the identity of the inventor (Rule 4.17(i))*
— *as to applicant's entitlement to apply for and be granted a patent (Rule 4.17(ii))*
— *of inventorship (Rule 4.17(iv))*

Published:

— *with international search report (Art. 21(3))*

document (T) between a user (U) and a recipient (V) is comprised of a transaction content (d) from the user (U) to the recipient (V) and a transaction content hash value (h(d)) signed with the private key (σ_U) of the user (U).

A method for generating a digital identity, a digital identity, a method for creating an electronic transaction document and an electronic transaction document

The present invention relates to the field of cryptographic techniques and database techniques, and particularly to the generation of a digital identity and creation of an electronic transaction document. The invention enables in a novel way the management of contracts and digital transactions in a cryptographically certified, secure, immutable and durable way.

The techniques 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

The method of the invention offers security and durability both for the content of contracts as well as for the bookkeeping of contracts in chains of transactions.

Technological Background / Basic Concepts

Laws of public cryptpgraphy

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

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

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

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 may remain anonymous or reveal, who U is as a person or an organization (authentication)

Encryption and decryption

Encryption of public documents

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

A document d can be encrypted and signed using the private key σ to compute σ(d), the result is just a long number C1. The encrypted version C1 of d can be decrypted again by using the corresponding public key π and computing π(σ(d)). This reflects the first basic mathematical law of public cryptography:

π (σ (d)) = d This law will be used later for encrypting and digitally signing a document.

Encryption of secret documents

The second basic mathematical law of public cryptography is the reverse of the first: **σ (π (d)) = d** This law is used for encrypting a document to hide its content.

Assume that d has beeen enciphered with the public key π of O resulting in **π(d)**. Therefore, it can only be deciphered and read by the owner of σ. Therefore, if π(d) is somehow obtained by anybody, e.g. a user U or even a hacker H by intercepting a message containing π(d), this is completely useless, since he cannot decipher it. To decipher π(d), he would need σ, but σ is by definition kept secret by its owner O.

Both laws together are the basic law of public cryptography:

π(σ(d)) = d = σ(π(d))

Digital Signature

An unciphered document d together with the ciphered version **σ (d)** denoted as **[d, σ(d)]** are a mathematical proof that d has not been changed, if the following property holds: **π (σ(d)) = d**

But if d has been changed to d1 resulting in **[d1, σ(d)]**, then computing **π (σ(d)) = d is unequal to d1** and therefore d1 cannot be the original d.

Since σ is secret, only the owner O of σ can have produced σ(d) with the property that π(σ(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 π(σ(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:

σ (π(d)) = d = π (σ(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 is practically impossible to compute its inverse $h^{-1}(d)$

Furthermore, for two distinct documents d1 and d2 it is extremely unlikely that h(d1) = h(d2). Therefore h(d) can also be used to prove that d has not been changed. But h(d) does not serve as a signature since anybody can compute h(d) for a publicly known function h.

Summary of the invention

Based on this, the invention proposes a method for generating a digital identity of a user with the features of claim 1, a digital indentity with the features of claim 3, a method for logging into a password protected computer application with the features of claim 4, a method for creating an electronic transaction document with the features of claim 5 as well as an electronic transaction document with the features of claim 7 and a method to manage an electronic transaction document on a database with the features of claim 9.

The invention offers a ryptographically secured, immutable, scalable, efficient management of digital contracts. This is called ***C-chain*** in the following**.**

Further features and embodiments of the invention will become apparent from the description and the accompanying drawings. In the drawings, Figures 1 to 8 show various examples and embodiments oft he invention.

It will be understood that the features mentioned above and those described hereinafter can be used not only in the combination specified but also in other combinations or on their own, without departing from the scope of the present invention.

The invention also covers a computer program with program coding means which are suitable for carrying out a process according to the invention as described above when the computer program is run on a suitable computer, or database management system. The computer program itself as well as stored on a computer-readable medium is claimed. The term computer is to be understood to cover any kind of computing device including personal computers, database management systems, smartphones, microcontrollers implemented on smart cards or any other devices used for data transaction.

A digital identity **cryptID** according to the invention is the pair **[π, σ(h(π))]**

For a particular user U his cryptID is denoted by the pair [π_U, σ_U (h(π_U))]

The cryptID of U can easily be generated by U and checked by another user V as follows:

1. compute the hashvalues v1 and v2 as $h(\pi_U)$ = v1 of the first component of the pair
2. decrypt $\sigma_U(h(\pi_U))$ of the second component of the pair by using the first component to compute $\pi_U(\sigma_U(h(\pi_U))) = h(\pi_U) = v2$
3. if v1 = v2 it is certain, that U ist he owner of π_U

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, see patent claim 2.

A cryptID is generated by U as follows:

1. generate a key pair $[\sigma_U, \pi_U]$ by using suitable software on a computer like on a smartphone, a tablet, a PC or a server.
2. Compute the hash function $h(\pi_U)$ which is easy
3. Digitally sign $h(\pi_U)$ by computing $\sigma_U(h(\pi_U))$ which is also easy
4. Form the pair $[\pi_U, \sigma_U(h(\pi_U))]$ as the cryptID of U
5. U can now present his cryptID to a communication partner V or simply publish it in the publicly available database UDB.

Using a digital identity of the invention, a login into other Computers with devices like a smartphone, computer, etc can be effected as follows:

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

1. The cryptID $[\pi_U, \sigma_U(h(\pi_U))]$ is used as the **login name**
2. A signed password **PWD** is generated on the device as follows:
   a. Choose a random number r from a sufficiently large range of numbers
   b. Sign r as $\sigma_U(r)$
   c. Use $[r, \sigma_U(r)]$ as the signed PWD
3. U now logs in with the pairs **$[[\pi_U, \sigma_U(h(\pi_U))], [r, \sigma_U(r)]]$**
4. This can even be simplified to log in with **$[\pi_U, [r, \sigma_U(r)]]$**

Since r is signed by U, nobody except U could have produced the PWD $[r, \sigma_U(r)]$. 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:

1. The pair [σ_U , π_U] is generated automatically by the device of the client (smartphone, tablet or PC)
2. U does not have to invent a new login name
3. U does not have to invent a password to obey security levels
4. U does not have to remember any login names and passwords
5. U does not need to store a copy of his passwords in a safe place as often recommended for logins
6. U avoids the danger of his password being stolen, since it is never visible, not even U himself needs to know his password
7. U can and should use different cryptIDs for the various applications in the Internet

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. To guard against steeling of passwords, a server easily keeps track that passwords are not reused. If such a password has 20 Bytes and if a user logs in 5 times a day, it would result in 100 B/day or 36 KB/year/user, a negligible datavolume.

In another embodiment, the random number r is generated by a web service and transmitted to U. In this embodiment, the server does not need to keep track of the used passwords.

Many users use the same login name and password to log into different computer systems or internet platforms. This is a dangerous habit: if for whatever reason the login data are lost or stolen, other platforms are compromised, too. Therefore, it is advisable, to use the cryptID with a different PWD for logging into different servers or WEB-services. Since cryptIDs and PWDs are generated automatically on the client device, the user does not even notice this.

With the conventional login technique, V could use the email adress of U and choose an arbitrary PWD for logins, but using a cryptID and a dynamically generated PWD, this guarantees, that the PWD was generated by U and not by V.

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. σ_U ([A, B]) is a certificate issued by user U. 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 σ_V ([V, π_v]) By this V certifies himself with his own signature σ_V that V and π_v belong together. We used the selfcertificate σ_U (h(π_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 daily life authentication of people happens by showing some ID card like a passport, a drivers licence, or simply the posession of a credit card. Authentication is usually performed by a third person checking certain biometric properties like the photograph in a passport or on the drivers license or simply the possession of a credit card, or in criminal investigations the fingerprint or a DNA sequence.

In the C-chain system we use the novel idea of a **cryptID** combined with **biometric authentication** to certify that a person is the owner of a public key. For this, U could post in the **UDB** a photo of his face like on a passport, but anybody else could easily post such a photo, too. It is much more secure that U records a video in which he reads the hash h(π_U) of his public key, while h(π_U) is shown simultaneously on the screen for immediate comparison. This biometric property would be extremely hard to fake. This video is then publishd in the UDB together with π_U and h(π_U). Of course, this video must be signed by U to prevent V from making a video and using the same h(π_U). Note that for this

proof we do not need to know σ_U (for σ_U must always be kept secret) nor do we need to know anything about the owner U. All we know is that U has the private key σ_U whoever U might be as a person or some other entity like an organization or a computer system.

In this way, anybody else – e.g. another user V – easily finds U and his data in UDB and can convince himself that π_U really belongs to U, because everything that V needs is to communicate with U is the public key of U. In this way U certifies himself, that π_U is his public key, even if the entity U remains anonymous. V may now use π_U to verify signatures of U or send secret messages to U.

Additional Information about U

In addition, the name, email adress, phone number, company url, social media url etc. of U could be stored in the UDB also, depending on the personal decision of U. Of course, these digital objects should be signed by U, too.

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 or to hide the real emission of cars. 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. Trust is their most important business asset. If e.g. a pizza service does not deliver the pizzas exactly as ordered or too late or not at all, a customer will not order again. If a table reserved in a restaurant is not available upon arrival of the guests, the guests will avoid this restaurant in the future.

Hackers denoted by H: They are trying to attack the C-chain system, e.g. in order to steal cryptIDs and private keys and to use them for shopping or for diverting money to their own accounts. But 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

If a logistics company like UPS or DLH is involved in delivering the package containing P, even more transactions are involved in the complete business process.

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 as follows:

[d , σ_U (h(d))] containing d as open data and the signature of the hash of d to assert the correctness of d. U is responsible that the content d is correct and follows the rules of the business involved.

Messages

Public messages

If U wants to send a message M to V containing the transaction T, this is extended by the sender and the recipient and now has the format:

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

The two public keys at the beginning of M determine the sender U and the recipient V uniquely. Other identification possibilities exist to determine sender and recipient, for example the digital identity of the recipient may be used if readily available. The sender of course may also use his or her digital identity fort hat purpose. For practical purposes and efficiency, the message may

contain additional data like the email adresses of U and V as well as their names in clear text.

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

Secret messages

If U wants to send a message M to V containing the transaction T, but wants to conceal the content of the message, this message has the format:

$$[\ \pi_U, \pi_V, [\pi_V(\alpha), \alpha(d), \sigma_U(h(d))\ ]\ ] = SM$$

Note: Here we use a symmetric key α to encrypt the content d, since private/public keys are not very efficient for encrypting long documents. Of course, we must communicate this key α secretely to V. For readability we abbreviate such a secret message in this format as SM.

Now the content d is encrypted and only V can read it, but ererybody can see, that U and V are communicating.

Hiding the sender of a message

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

$$[\pi_V(\pi_U), \pi_V, [\pi_V(\alpha), \alpha(d), \sigma_U(h(d))\ ]\ ] = HM$$

We abbreviate a message in this format as HM.

Now only V can see the sender U. Such messages are required if e.g. a company wants to bid for a request of proposal for a project by V, since the information, who the other bidders are, should be secret.

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 some obvious rules are:

1. The product number of P must be correct
2. the acknowledgement must contain the correct product number and price
3. V must send the correct product and not something else

4. In the bill the price of the product and the account number of V must be correct
5. U must pay the correct price and copy the account number of V correctly
6. B must check the existence of the bank account of V and transfer the right amount

Every transaction must be formulated and executed correctly. In digital business processes many transaction steps are performed by software programs or at least accompanied by software programs and database transactions.

In principle these rules can be obeyed by the user who formulates a transaction and checked for correctness by the recipient of the transaction. Therefore, instead of manual formulation and checking of such rules, the rules are programmed into the software for each individual transaction. ***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 redundant 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.

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:

1. The transaction T received by TDBMS is checked for **proper certification** first. Therefore, T authored by U and targeted for V has the following form in C-chain: **$T = [\pi_U, \pi_V, d, \sigma_U (h(d))]$** Note that in this form T is certified by U. In this form the complete transaction is publicly visible. Public visibility is desired for certain forms of transactions, e.g. for the standard application of blockchains or for bidding in public auctions.
2. Before TDBMS appends T to the end of the chain, **TDBMS must check** that T is certified i.e. it computes using the various components of T:
   a. v1 = h(d)
   b. v2 = π_U (σ_U (h(d)))
   c. If v1 = v2 then T has been certified by U and can be booked, otherwise TDBMS rejects T and optionally informs the sender π_U, that T is not correct
3. **Certification of T by the system TDBMS** with its own private key σ_S: After checking the proper certification by U, TDBMS now also signs T. Thereby T can no longer be changed or replaced, not even by U. Without the certification by TDBMS, U could decrypt T, modify T (e.g. to change the price of a sale) and sign T again. The additional certification by TDBMS

makes this impossible. Changing T would only be possible if U and TDBMS form a complott and cooperate to replace T. But TDBMS will not do this by definition. In addition, the recipient V of a transaction would discover the misbehaviour.

4. 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 form [n+1, $h(T_n)$, T] and sign it by σ_S of TDBMS, i.e. store σ_S [n+1, $h(T_n)$, T] in TDBMS. $h(T_n)$ as part of T_{n+1} makes sure, that T_{n+1} was added to the chain bei TDBMS and not by a hacker H. Note that T is not appended by U himself, but only indirectly by TDBMS.
5. The TC – which is stored also locally by U - is synchronized, so that U can check immediately, that his transaction was properly booked
6. The TC stored locally by V is synchronized by a push notification of TDBMS or as soon as V turns on his client device.

If TDBMS is a public database, then π of TDBMS is known or can be looked up in UDB and everybody, in particuar U and V, can check that T has been booked properly. Furthermore, U and V can keep local copies of the transaction chains in which they are involved as agents. So they have additional undeniable proof of what happened in the business process.

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). This would be an additional fortification of TC, since now the last element T_{n+1} of TC could not be replaced by another T_{n+1}

Datarecord in TDBMS:

The detailed structure of the data record for T_{n+1} in TC then looks as follows:

[n+1, $h(T_n)$, σ_S (T)] = [n+1, $h(T_n)$, σ_S ([π_U, π_V, d, σ_U (h(d)))]]

For simplicity and to keep the notation readable, we used σ_S **(T)**, but of course every component of T could be signed individually.

Secret transactions (SM) or with a hidden sender (HM) are booked analogously.

Creating a new Chain: of course, a user must be able to create a new chain. 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: **[0, 0,** σ_S **([** π_U**,** π_S **, d,** σ_U **(h(d)))]]**

Proof of correctness of the Protocol

We argue that the protocol fulfills all the requirements for transaction chains in the order as they were stated above:

1. Property 1 (**certification**) is guaranteed by TDBMS in step 1 and 2 of the protocol, since all transactions T_k with k <= n+1 are signed by the author and the σ_S of TDBMS. TDBMS has of course its own key pair [σ_S , π_S] and signs all appends to TC. T_k could only be changed by TDBMS and U together in a complott, since only TDBMS knows σ_S. But TDBMS is trusted by definition, TDBMS will not do this. In addition at least U and V have copies of TC and would recognize, that TDBMS is suddenly cheating, i.e. maybe hacked by H, and they would not accept the modified TC. For further defenses against hacking see poin 5 and the paper on SystemArchitecture and the concept of public auditors, who can check the public database TDB which contains all transaction chains.
2. The **immutability** of T is achieved by Step 3.
3. The **strictness** of TC is enforced by steps 3 and 4 in the protocol and by using the classical techniques of databases for the serialization of transactions. Including $h(T_n)$ as one component of T_{n+1} additionally asserts the strictness of TC
4. This property of **appends only** is enforced by TDBMS in step 4
5. The **durability** of TC is guaranteed by storing the database redundantly on several hard disks, several local servers or even in several independent clouds, depending on the requirements of the application. Storing in several clouds would be suitable even against hacks or terrorist attacks.
6. **Final settlement** is guaranteed by Step 4c

Further essential properties of C-chain

In addition to correctness of the protocol of C-chain processing, there are other important properties:

1. Digital identities cryptID [π, σ (h (π))] are stored in a public database UDB and can easily be found and verified.
2. To increase security, U could expand his pure cryptID optionally by a photo or video for authentication, his name, email address, his company, 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. In such a way, the cryptID with the public key and/or authenticity of U must be checked by V only once and very conveniently
4. Perfect scalability
5. Very fast processing
6. Immediate final settlement
7. Etc, there is a summary of the comparison with blockchains and the advantages of C-chain over the blockchain technology in another paper

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.

Example Banking:

- **Agents**
  - Owner O of an account
  - Bank B
  - Recipient R of a transfer
- **Types of transactions**
  - Show
  - Withdraw

- Transfer

- **Parameters**
  - accountNr
  - amount
  - bill

Formats for Transactions

Show **[π_0, π_B, [π_B (α), α(d), σ_O (h(d))]]** where d = [show, accountNr]

Withdraw **[π_0, π_B, [π_B (α), α(d), σ_O (h(d))]]** where d = [withdraw, accountNr, amount] B must be able to check, that the account is not overdrawn

Transfer **[π_0, π_B, [[π_B (α), α(d_B), σ_O (h(d_B))], [π_R (β), β(d_R), σ_O (h(d_R))]]** where d_B = [transfer, $accountNr_O$, $accountNr_R$, amount] d_R = [transfer, amount, bill] B does not need to see the bill, R does not need to see the accountNr of O

Example medical Prescription:

- **Agents**
  - Doctor D
  - Patient P
  - Pharmacy A
- **Types of transactions**
  - Prescribe: D prescribes for P
  - Present: P presents prescription to A
  - Deliver: A delivers medication M and bill to P
- **Parameters**
  - Medication: M, the pharmaceutical name of a medication
  - Amount : A, e.g. the number and strength of pills
  - Prescription number: PN = **[π_D , π_P , sequenceNr]** a unique prescription number by a certain doctor for a certain patient to prevent multiple spending of a prescription at different pharmacies

Formats for Transactions

Prescribe $[\pi_D, \pi_P, [\pi_P(\alpha), \pi_A(\alpha), \alpha(d), \sigma_D(h(d))]]$ where
d = [prescribe, PN, M, A] here the patient and the pharmacy are able to see the details of the prescription

Present $[\pi_P, \pi_A, [\pi_P(\alpha), \pi_A(\alpha), \alpha(d), \sigma_P(h(d))]]$ where
d = [present, PN, M, A] here the patient and the pharmacy are able to see the details of the prescription

Deliver $[\pi_A, \pi_P, \sigma_A(h(d))]]$ where d = [deliver, PN, date] this transaction is public and all pharmacies can see that the prescription has been delivered to prevent double spending

Applications of the above-described invention „C-chain“ are disclosed in the following:

Summary

In business processes several agents can be involved, but in the single transactions typically two partners are active, the author and sender A of the transaction and the recipient B.

Each agent

1. may be **trusted** and play fair (+), usually because it is in his business interest to be trusted. In normal situations he will not cheat. In this category are banks, notaries, shopping centers, restaurants, etc. By far most business processes rely on such trust, but their business processes can be streamlined and made more secure by C-chain
2. an agent may be **untrusted** and is not necessarily fair (-) and might cheat if it is to his advantage. This typically happens, if A or B conduct business only once, e.g. selling an item on the flea market, selling a piece of art, paying with forfeit money in cash, transferring an asset several times but delivering only once or not at all, selling a low quality or defect used product on ebay, misstating the quality of the sold item, etc.

We consider all four combinations and give typical examples for situations.

Case 1: A- and B-

1. Dealing with drugs or weapons anonymously in the darknet,
2. selling on a flea market,
3. A selling a used car with manipulated mileage and B paying in cash with forfeit money

Case 2: A- and B+

minor product or service quality, missing delivery or denial of a service

1. Selling a house without having a title on it or with invisible damages, like a leaking roof,
2. selling a used car with manipulated mileage,
3. Trying to sell an item several times (double spending)
4. Denial of delivering a sold item, e.g. somebody sells an antique chair, but receives a much better offer shortly thereafter, and if the first buyer wants to pick up the item the seller denies having made the sale
5. A taxi company not sending the taxi because it was needed for a longer and more lucrative trip, the customer waits in vain
6. A pizza shop delivering the wrong pizza

Case 3: A+ and B- this happens frequently, but often not intensionally

1. A customer not waiting for an ordered taxi
2. A customer reserving a table in a restaurant, but not showing up
3. A customer ordering heating oil by phone to be delivered in two weeks. In the meantime the price drops and the customer denies having ordered the oil when it arrives and refuses to accept it. The dealer has no proof of the order, unless he taped the phonecall, which is illegal.
4. B takes a credit, spends the money and goes into bankruptcy

Against all examples of case 3, A has developed certain strategies to avoid damage as far as possible:

- In minor cases A takes the risk of lost business or insists on prepayment like for expensive opera tickets
- In some cases like for reserved movie tickets the seller tries to recover part of the lost business by selling the ticket if it is not picked up 30 min before the show
- In major cases the seller or a bank giving a loan insists on some security like on the title on a house as protection against a credit default or uses escrow transactions

Case 4: A+ and B+

This is probably the most frequent situation **except for payments**. Many people would like to spend their money several times, if they could do so easily without being caught.

Examples of Applications of C-Chain

Login to Computers and WEB-Services

1. Using **cryptIDs** for login and PWD (both are generated automatically and stored on the client) is much more convenient and much more secure than the conventional method with login name and password
2. Management of digital identities cryptID and public keys on a pubic **database UDB**
3. The conventional method of obtaining MIME **certificates** or via the PKI infrastructure is much more complicated and lengthy than via the database UDB and biometric authentication.

Financial services

1. Convenient and more secure (no forfeits) **substitute for cash payments** without a change problem.
2. **Simplified electronic banking**: bank transfer via C-chain is significantly simplified by the cryotographic signature. In particular, it requires no infrastructure except a smartphone. In contrast, conventional electronic banking requires a complex and expensive infrastructure:
    a. A computer with a browser
    b. A fast internet connection
    c. Conventional login
    d. A bank card
    e. A CHIP TAN generator
3. **Substitute for EC-Card payments**: money transfer authorized by C-chain is much simpler than the present method with EC-card, card reader, PIN or manual signature and the followup processing of the debit order

4. **Simplified debit orders:** conventional debit orders are complex. With C-chain a debit order with a digital signature can instantly be sent to the payee. This method can be used by C-chain to send money to another person even if the bank does not support C-chain.
5. **Transferring cash:** the simple *cash transfer* system KWITT propagated by German banks requires the interaction of two agents: the payer and the payee, who must respond to an SMS, open a browser on his smartphone, enter his name and copy his lengthy error prone IBAN from his EC card. With C-chain this is much simpler
6. **Transferring money to foreign countries:** today this is extremely lengthy, complicated and expensive, e.g. for fugitives from the middle East or from Africa. With C-chain this becomes much easier, cheaper and faster and can be performed even by private people with significantly reduced interaction with banks
7. **Payments in shopping centers, supermarkets, restaurants, gas stations, etc:** just read a QR code and push the payment button.

Replacement of all chip cards

In principle C-chain is suitable as a very convenient, more secure and at the same time cheap replacement for all chip cards

1. **EC card**, see details above
2. **Membership cards** for clubs, e.g. ADAC, PayBack card of Kaufhof Galerie, premium member cards of airlines, etc. In this case, the club digitally signs a document with his own logo etc which is shown on and presented by the smartPhone of the member. Everybody can check the signature of the club and thereby verify the membership. No infrastructure for card readers is needed, in addition it opens up a communication channel between Club and members with highly improved customer relationship.
3. **Health Insurace cards** proving insurance coverage

Replacement of physical keys including RFID chips

Highly secured and confidential medical and legal information

1. **Patient provision**: This is extremely sensitive and confidential information. Therefore, patient provisions are often deposited with public notaries. But medical doctors need quick and reliable information about patient provisions. If an accident victim dies, medical decisions

about organ transplants must be absolutely confidential and must be made in extremely short time. But to disclose this information to a medical doctor, a doctor must authenticate himself with his eHBA (elektronischer Heilberufe Ausweis), a chip card, which all doctors should have. But this card is very expensive and out of the more than 300.000 doctors in Germany, presently only about 6.000 have the card. An eHBA based on C-chain would have many advantages like: speed, security, ease of use.

2. **Patient Record**: Germany has been trying in vain for many years and at great public expense (1.7 billion € so far), to establish such a system. A patient record based on C-chain woud be much more acceptible to the public and much cheaper, and it would have many additional advantages, e.g. parents could have the health record of their children on their smartphones and coud send them selectively to a doctor, to emergency services or a hospital. Even if the parents are at work and cannot be present with their children, they could send this information from their smartphones. This would also include the insurance card for their children.

Supply Chain Management

Supply chains are often long, transnational, but they must be managed quickly and highly securely and reliably. Supply chains are often used as a prime example of blockchain technology, but C-chain is a much better alternative. If anything goes wrong, costs can be extremely high (interrupted production lines) and the question arises, who is liable fort he damage. Such processes can be streamlined and made much more secure, if all transaction steps in the chain are cryptographically secured and can be followed by making the chain visible and checkable to all partners. Since signatures cannot be denied, the immediately identified.

Car sharing

This is gaining popularity very quickly, but it is still rather inconvenient: Chip cards are needed and safes containing the car key must be opened with a particular code. The following C-chain process would simplify car sharing and car renting:

Here is the transaction chain for renting a car:

Example of a car rental process with 3 agents: user U, car rental agency V and car A

The transaction chain: *The content of the transaction is shown in Italics.*

1. **U → V** **U gives V the right, to deduct up to 300 € from bank account of U or transfers € to V by bank transfer**
2. **U → V** *I need a rental car+details (my age, license #, where needed)*
3. **V → U** *here are several offers, choose your preference*
4. **U → V** *I book the following offer ...*
5. **V → A** *unblock the engine*
6. **A → V** *engine was started, A can transmit details to V*
7. **U → V** *I returned A to location X*
8. **V → A** *block the engine and transmit present location*

Reservations and sales of tickets

for public and private transportation, theaters, concerts, etc

Hotel Reservations and Payments

Supermarket Payments and shopping centers

When paying groceries in a supermarket, many people try to find the correct amount of cash in their wallets to make it lighter. This wastes a lot of time fort he clerks at the cash register. This payment process could be sped up by a C-chain solution. In addition to saving cost, this would open the path for more intense customer care.

Company Cafeterias and Employee Cards

Many companies have employee cards used for access to their premisies, to restricted areas, paying in the cafereria, and for additional services. These cards could be replaced by a C-chain solution.

Certified emails

Insurances

Microinsurances are a big market especially in third world countries. But sales and damage regulations by insurance agents is by far too expensive. With C-chain technology this market could be opened

Electronic prescriptions

In Germany alone about 700 million prescriptions are handled annually with antiquated partially manual business processes on paper involving doctors, patients, pharmacies and insurances. This process could be highly automated by C-chain technology.

Example of a Bank Account

Example Banking:

- **Agents**
  - Owner O of an account
  - Bank B
  - Recipient R of a transfer
- **Types of transactions**
  - show
  - withdraw
  - transfer this is the order of the account owner to his bank to transfer the money
  - deposit this is the transaction of the bank to deposit the money in the account of R
- **Parameters**
  - accountNr
  - amount
  - bill

- **Chains**
  - Every User has a chain for his account corresponding to a conventional bank account, accoountChain of O and accoountChain of R

Formats for Transactions

Show **[π_0, π_B, [π_B (α), α(d), σ_O (h(d))]]** where d = [show, accountNr]

Withdraw **[π_0, π_B, [π_B (α), α(d), σ_O (h(d))]]** where d = [withdraw, accountNr, amount] B must be able to check, that the account is not overdrawn

Transfer **[π_0, π_B, [[π_B (α), $\alpha(d_B)$, σ_O (h(d_B))], [π_R (β), $\beta(d_R)$, σ_O (h(d_R))]]** where d_B = [transfer, $accountNr_O$, $accountNr_R$, amount]
d_R = [transfer, amount, bill] B does not need to see the bill, R does not need to see the accountNr of O

The transaction chain:

1. **O → B** ***show*** *my account balance*
   a. The balance could always be shown as part of the last transaction booked as the end of the chain. Then O could also see the balance in the copy of his own transaction chain, which is stored locally on his device
2. **O → B*transfer*** *17 € from my account to R*
   a. Now B has the signature of O for carrying out this transfer. Note that this transaction is booked in the accountChain of O
3. **B → O** **deposit** *ok, I will deposit 17 € in the account of R*
   a. This transaction is booked in the chain of O. Now O has the signature of B, that B is depositing the 17 € to R.
4. **B → R** *this is a deposit of 17 € in your account*
   a. This transaction is booked in the chain of R. Now R has the signature of B, that 17 € were deposited in his account.

In this example, the booking process shows and proves, that B is really carrying out the order of O. Whether this is possible or not depends on the specific application. If physical goods are transferred in the application like in a supply chain or by the delivery of a product by a mail-order company as a

package, then the courier service hands out the package only if the recipient R makes a transaction, which is the classical handwritten signature on a receit.

Details of a transaction *transfer* in example 2 above

Actions of O

1. O opens the client program for C-Chain, e.g. the App called **Chain** on his smartphone or a program **Chain** on his PC
2. as a result Chain shows a list of all the transaction chains to which O is admitted as an agent, in particular the account chain AC_O of O, but also others
3. O selects AC_O from this list by a click
4. Now the App shows a list of all transaction types which O is allowed to perform with this transaction chain, e.g. **show, withdraw, transfer, deposit**
5. From this list O selects what he wants to do, here transfer
6. Now Chain presents a specific form for the transaction type transfer.
7. This form has several fields which must be filled out, e.g.
    a. A **list of several cryptID**, which are already known to and stored in Chain. We assume that the cryptID of R is in this list, since R could be a supermarket with $cryptID_S$, in which U buys groceries regularly. How U gets this cryptID is described below, since U must do this only once
    b. A **field for the transaction document d**. This could be a subform with several fields This form would be similar to the form used in electronic banking today. But for simplicity we assume that this field is simply a text field
8. O now fills out this form:
    a. O selects a cryptID for the recipient of this transaction simply by clicking it, here the $cryptID_B$ of the bank B
    b. O fills out d e.g. with the following text: ***please, transfer 17 € to the account with IBAN number DE73 ... of $cryptID_R$*** . If d is a subform, some of these fields would already be filled out like in electronic banking today.

9. O clicks the button **send.** Before sending, Chain performs all required signatures etc and builds the transaction document as described before. All this happens automatically and the user O does not see any of that.

This Transaction T is sent to the booking database system TDBMS via an arbitrary communication channel like https or email, etc. TDBMS now books this transaction in the transaction chain AC_O. The bank B is notified by a push message or synchronization, that a new transaction has arrived for B.

B now performs the following actions

1. B analyses the content d of the received transaction T, checks proper signatures and extracts from it ***17 €, IBAN number, cryptID$_R$***
2. B formulates a new transaction T2 for O to confirm, that B will carry out the transfer of the money to R. T now has the commitment of B
3. B moves the money from the account of O to the account of R and formulates a transaction T3 informing R that the money has been deposited in the account of R. R sees this immediately and now has proof, that B transferred the money from O to the account of R.

Getting the cryptID of another User R

There are several alternatives

1. ***cryptID$_S$*** could be displayed as a QR code at the cash register of the supermarket and is photogtaphed by Chain once and stored locally in Chain
2. ***cryptID$_R$*** can always be found in the UDB. For this purpose Chain offers convenient search and verification support
3. ***note that this is done only once for an agent R with which U wants to interact***

Claims

1. A method for generating a digital identity (cryptID) of a user (U) for user authentication in electronic transactions, the user (U) being in possession of a cryptographic key pair (σ_U, π_U) comprising a public key (π_U) and a private key (σ_U), the method comprising the following steps:
   - computing a hash function of the public key (π_U) thus generating a public key hash value ($h(\pi_U)$);
   - digitally signing the public key hash value ($h(\pi_U)$) with the private key (σ_U) thus generating a signed hash value ($\sigma_U(h(\pi_U))$;
   - establishing the digital user identity (cryptID) to be the pair consisting of the public key (π_U) and the signed hash value ($\sigma_U(h(\pi_U))$.

2. The method according to claim 1, comprising the additional step of sharing the digital user identity (cryptID) by electronically sending the digital user identity (cryptID) to another user (V) and/or uploading the digital user identity (cryptID) onto a publicly accessible database (UDB).

3. The method according to claim 1 or 2, wherein the digital user identity (cryptID) is combined with a biometric property of the user (U).

4. The method according to claim 3, wherein the biometric property of the user (U) is a video recording of the user (U) reading out the content of the public key hash value ($h(\pi_U)$).

5. The method according to claim 4, wherein the content of the public key hash value ($h(\pi_U)$) is simultaneously displayed.

6. The method according to claim 4 or 5, wherein the video recording is signed with the private key (σ_U).

7. A digital identity (cryptID) of a user (U) for user authentication in electronic transactions, the user (U) being in possession of a cryptographic key pair (σ_U, π_U) comprising a public key (π_U) and a private key (σ_U), the digital identity (cryptID)

being comprised of the public key (π_U) of the user (U) and a public key hash value ($h(\pi_U)$) digitally signed ($\sigma_U(h(\pi_U))$ with the private key (σ_U) of the user (U).

8. The digital identity (cryptID) according to claim 7, combined with a biometric property of the user (U).

9. The digital identity (cryptID) according to claim 8, wherein the biometric property of the user (U) is a video recording of the user (U) reading out the content of the public key hash value ($h(\pi_U)$).

10. The digital identity (cryptID) according to claim 9, wherein the content of the public key hash value ($h(\pi_U)$) is simultaneously displayed.

11. The digital identity (cryptID) according to claim 9 or 10, wherein the video recording is signed with the private key (σ_U).

12. A method for logging into a password protected computer application using a digital identity (cryptID) according to claim 7 as a login name and a signed password generated by the steps of
- choosing a random number (r);
- digitally signing the random number thus generating a signed random number ($\sigma_U(r)$);
- establishing the signed password to be the pair ([r, (σ_U(r)] consisting of the random number and the signed random number.

13. A method for creating an electronic transaction document (T) between a user (U) and a recipient (V), both the user (U) and the recipient (V) each being in possession of a cryptographic key pair comprising a public key (π_U; π_V) and a private key (σ_U; σ_V), the method comprising the following steps:
- providing a transaction content (d) from the user (U) to the recipient (V);
- computing a hash function of the transaction content (d) thus generating a transaction content hash value (h(d));
- signing the transaction content hash value (h(d)) with the private key (σ_U) of the user (U);

- establishing the transaction document (T) to be the pair consisting of the transaction content (d) and the signed transaction content hash value ($\sigma_U(h(d))$).

14. The method of claim 13, wherein the transaction document (T) further comprises the public key (π_U) of the user (U) and the public key (π_V) of the recipient (V).

15. The method of claim 13 or 14, wherein the transaction content (d) in the transaction document (T) is encrypted using a symmetric encryption key.

16. The method of claim 15, wherein the transaction document (T) further comprises the symmetric encryption key encrypted by means of the public key (π_V) of the recipient (V).

17. The method of any one of the claims 14 to 16, wherein the public key of the user (U) is encrypted by means of the public key (π_V) of the recipient (V).

18. An electronic transaction document (T) between a user (U) and a recipient (V), both the user (U) and the recipient (V) each being in possession of a cryptographic key pair comprising a public key (π_U; π_V) and a private key (σ_U; σ_V), the electronic transaction document (T) being comprised of a transaction content (d) from the user (U) to the recipient (V) and a transaction content hash value (h(d)) signed with the private key (σ_U) of the user (U).

19. The electronic transaction document (T) according to claim 18, wherein the transaction document (T) further comprises the public key (π_U) of the user (U) and the public key (π_V) of the recipient (V).

20. The electronic transaction document (T) of claim 18 or 19, wherein the transaction content (d) in the transaction document (T) is encrypted using a symmetric encryption key.

21. The electronic transaction document (T) of claim 20, wherein the transaction document (T) further comprises the symmetric encryption key encrypted by means of the public key (π_V) of the recipient (V).

22. The electronic transaction document (T) of any one of the claims 19 to 21, wherein the public key (π_U) of the user (U) is encrypted by means of the public key (π_V) of the recipient (V).

23. A method to manage an electronic transaction document (T) according to any one of claims 18 to 22 in a transaction chain (TC) on a database (TDB), the method comprising the following steps:
- receiving an electronic transaction document (T) to be archived;
- checking certification of the electronic transaction document (T);
- certifying the electronic transaction document (T) with a database key (σ_S);
- appending the electronic transaction document (T) to the transaction chain (TC).

24. The method of claim 23, wherein the step of appending comprises
- reading the previous last entered electronic transaction document (T) contained in the transaction chain (TC) and computing the hash value thereof,
- identifying the position (n) of the last entered electronic transaction (T_n) and incrementing it by one;
- creating a triple of the incremented position (n+1), the hash value of the previous last entered electronic transaction (T_n) and the electronic transaction document (T);
- signing the triple with the data base key (σ_S);
- storing the signed triple as new last electronic transaction (T_{n+1}).

25. A database system (TDBMS) implementing a method according to claim 23 or 24.

26. A method of effecting an electronic transaction using a digital identity (cryptID) according to any one of claims 7 to 11 and using a transaction chain (TC) on a

database system according to claim 25, under the control of a user or client system, comprising the following steps:

- displaying at least one transaction chain (TC, AC_0) to a user (U, O) which the user (U, O) is authorized to use;
- receiving a first selection input from the user (U, O) which transaction chain (TC, AC_0) is to be used;
- displaying a list of possible or preselected transaction types to the user;
- receiving a second selection input from the user (U, O) which transaction type is to be used;
- displaying a transaction type specific electronic form to the user;
- receiving the filled-in form from the user containing at least a public key or a digital identity ($cryptID_B$) of a recipient (V, R, B) and a transaction content (d);
- creating an electronic transaction document (T) according to any one of claims 18 to 22 based on the filled-in form;
- sending the electronic transaction document to the database system.

27. A method to operate a database to manage an electronic transaction, under the control of a database system, comprising the following steps:

- receiving an electronic transaction document created and sent according to claim 26;
- appending the electronic transaction document according claim 24 to a transaction chain specified in the electronic transaction document;
- forwarding the electronic transaction document to the recipient (V, R, B).

28. A method to process an electronic transaction using a digital identity (cryptID) according to any one of claims 7 to 11 and using a transaction chain (TC) on a database system according to claim 25, under the control of a recipient system, comprising the following steps:

- receiving, from the database system, a certified electronic transaction document already appended to a transaction chain according to claim 24;
- acknowledging the transaction content (d) contained in the transaction document by creating a new transaction document (T2) according to any one of claims 18 to 22;
- sending the new transaction document (T2) to the database system for appendage to the transaction chain;
- processing the transaction content (d) contained in the transaction document.

29. A computer system implementing a method according to any one of claims 26 to 28, the computer system being a host computer, a server system, a portable computing device, an information and communication device, a database system, or the like.

30. A computer program product with a computer-readable medium and a computer program stored on the computer-readable medium with program coding means which are suitable for carrying out a method according to any one of claims 1 to 6, 12 to 17, 23 to 24, or 26 to 28 when the computer program is run on a computer, smartphone, database system or any other suitable computing device or computer system according to claim 29.

31. A computer program with program coding means which are suitable for carrying out a method according to any one of claims 1 to 6, 12 to 17, 23 to 24, or 26 to 28 when the computer program is run on a computer, smartphone, database system or any other suitable computing device or computer system according to claim 29.

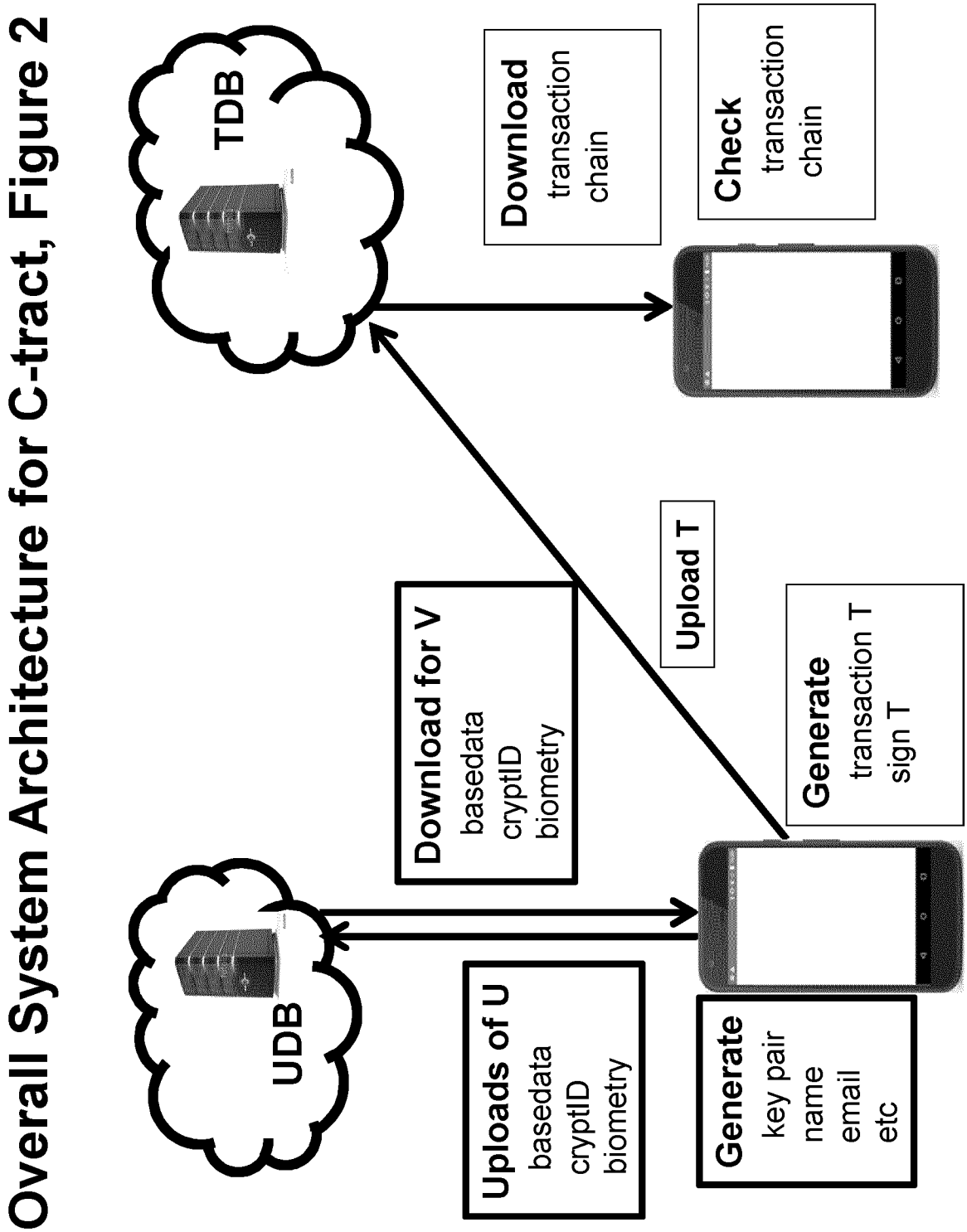
32. A computer-readable medium with a computer program stored thereon, the computer program comprising program coding means which are suitable for carrying out a method according to any one of claims 1 to 6, 12 to 17, 23 to 24, or 26 to 28 when the computer program is run on a computer, smartphone, database system or any other suitable computing device or computer system according to claim 29.

System Components of C-tract, Figure 1

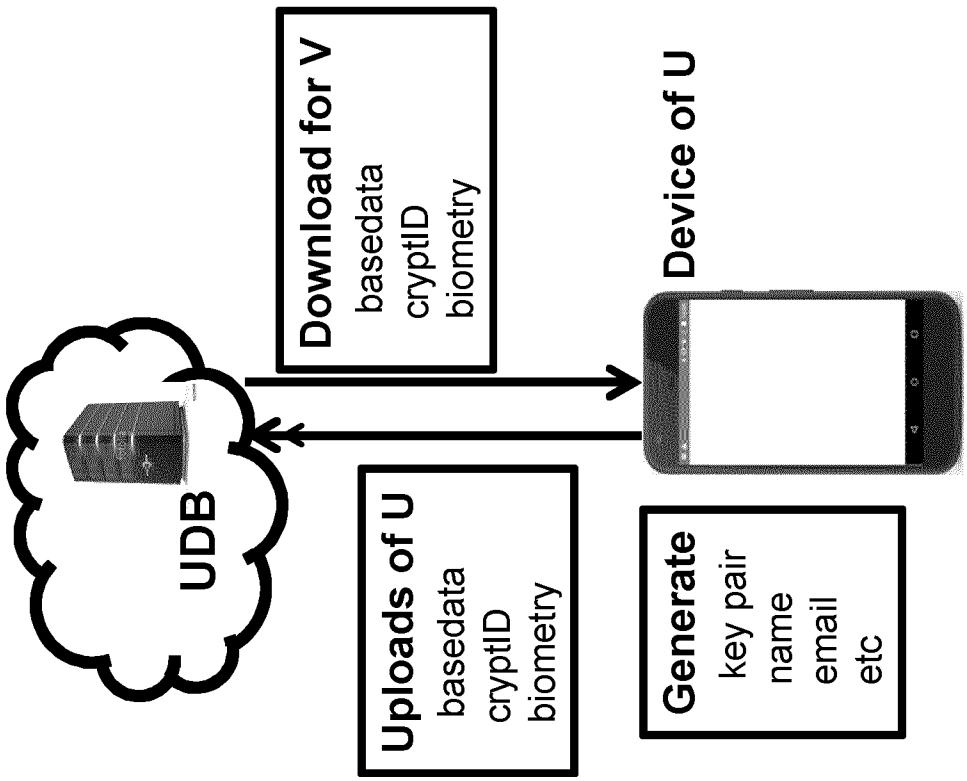
The User Database **UDB** plus software:
management system UDBMS
running somewhere as local server
or in the cloud

The Transaction Database **TDB** plus software:
management system TDBMS

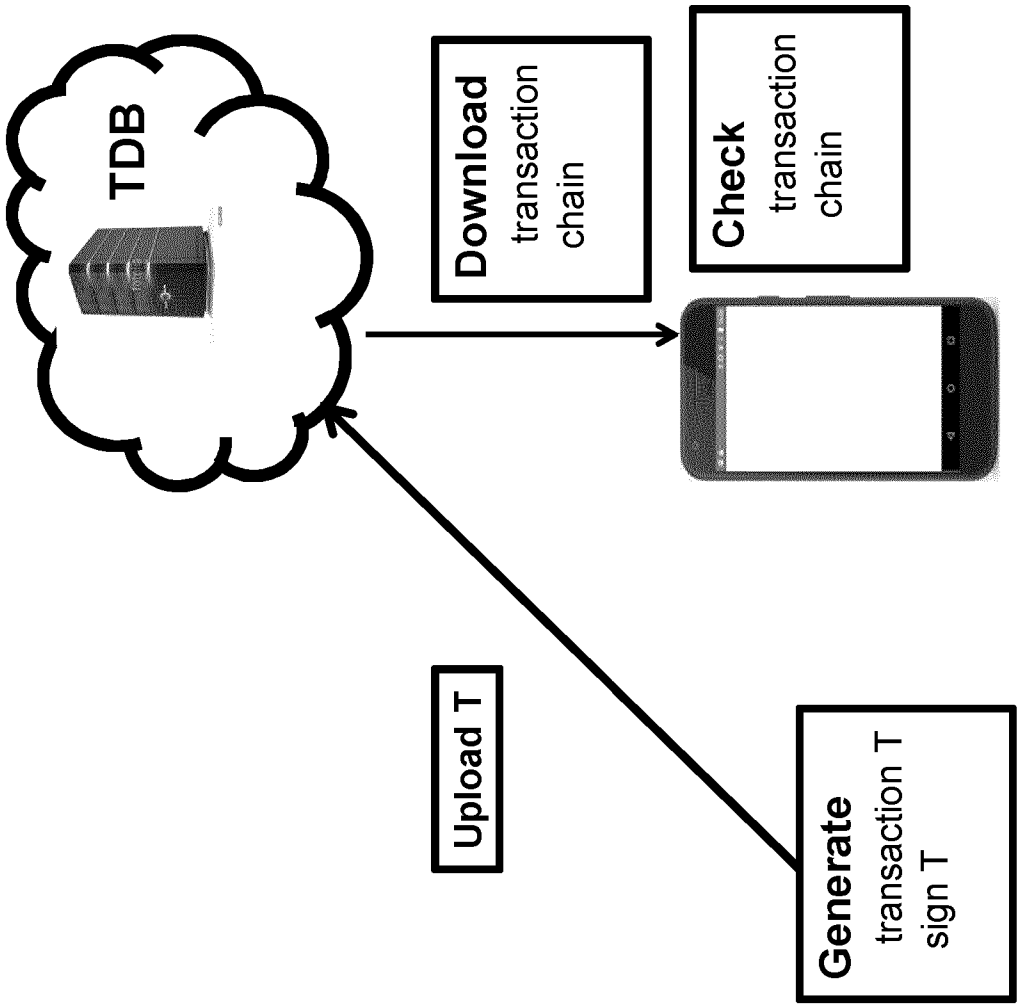
Client devices like
smartphones, tablets PCs
with Apps and software for
- generating key pairs
- generating transactions
- locally storing chains
- checking chains


Overall System Architecture for C-tract, Figure 2
TDB
UDB
Download transaction chain
Check transaction chain
Download for V basedata cryptID biometry
Upload T
Generate transaction T sign T
Uploads of U basedata cryptID biometry
Generate key pair name email etc

Interaction between U and UDB, Figure 3



Interaction between U and TDB, Figure 4



Defense against lost Key, Figure 5

1. Copy key pair to an external USB stick and deposit it in bank safe
2. Copy key pair into a file protected by PWD on PC
3. Copy into a file protected by PWD in the cloud
4. App has a function export/import key pair

Defense against Hackers H, Figure 6

1. **Key logger** virus is useless, since user never enters a PWD or his private key via the keyboard
2. **Man in the middle** is useless because of https
3. **Hacking UDB** this DB is open anyway and modifying it would require the private key of a user, since all entries in UDB should be signed
4. **Hacking TDB**
    1. Read access to TDB and chains is open
    2. Write access to TDB requires the key of TDBMS, for defense against stealing the private key, see next slide
    3. Changing the TC: for extremely sensitive applications use 4 or k eye principle for multiple signing of the entries in TC
    4. Changing the content of a single transaction requires key of author and of TDBMS, would corrupt the TC and would be noticed quickly
    5. Audit the TC and discover irregularities

Defense against stolen PWD or Key, Figure 7

1. **Stolen PWD**
    1. Login happens now with [cryptID, [random r, σ(r)]]
    2. Use **once only logins** with new r for every login
    3. [random r, σ(r)] functions as PWD, but this PWD is not stored anywhere, not on the server and not on the client, it is only generated dynamically within the App
2. **Stolen private key, extremely difficult**
    1. The private key is stored encrypted in the key chain of the device
    2. Therefore, the private key is only visible in the code of the running App
    3. To steal it would require manipulation of the code of the App and installation of the manipulated App
    4. Audit transaction chains to discover irregularities, this can be done by a separate audit system and it is done by every user for his transaction chains

SUBSTITUTE SHEET (RULE 26)

Dynamic Keys, Figure 8

For extremely sensitive applications dynamically changing privat keys can be used as follows:

1. **Change the key pair:** It is easy to change the key pair dynamically at arbitrary intervals, e.g. every day or hour
2. Then **destroy** the old private key, therefore it can no longer be stolen, it simply no longer exists anywhere
3. Old signatures remain valid and can be checked with the old public keys, which remain in the UDB
4. This results in chains of public keys (private keys are destroyed). Changing keys every hour causes 24x365 = 8760 remaining public keys or about 5 MB of data volume per year

Dynamic keys seem impossible with the presently prevailing technique of chip cards for digital signatures

INTERNATIONAL SEARCH REPORT

International application No
PCT/EP2018/075874

A. CLASSIFICATION OF SUBJECT MATTER
INV. G06F21/32
ADD.

According to International Patent Classification (IPC) or to both national classification and IPC

B. FIELDS SEARCHED

Minimum documentation searched (classification system followed by classification symbols)
H04L G06F

Documentation searched other than minimum documentation to the extent that such documents are included in the fields searched

Electronic data base consulted during the international search (name of data base and, where practicable, search terms used)
EPO-Internal, WPI Data

C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
X	Rudolf Bayer: "C-chain: a system for managing public and private ledgers, an alternative to blockchain", , 15 September 2017 (2017-09-15), XP055534286, Retrieved from the Internet: URL:https://cchain.transaction.de/pdf/C-chain-Scient.pdf [retrieved on 2018-12-13] page 2 - page 4	1-32
A	US 2016/328713 A1 (EBRAHIMI ARMIN [US]) 10 November 2016 (2016-11-10) paragraph [0041]	1-32
A	US 2007/016785 A1 (GUAY YANNICK [CA] ET AL) 18 January 2007 (2007-01-18) paragraph [0022]	1-32

☐ Further documents are listed in the continuation of Box C.

☒ See patent family annex.

* Special categories of cited documents :

"A" document defining the general state of the art which is not considered to be of particular relevance

"E" earlier application or patent but published on or after the international filing date

"L" document which may throw doubts on priority claim(s) or which is cited to establish the publication date of another citation or other special reason (as specified)

"O" document referring to an oral disclosure, use, exhibition or other means

"P" document published prior to the international filing date but later than the priority date claimed

"T" later document published after the international filing date or priority date and not in conflict with the application but cited to understand the principle or theory underlying the invention

"X" document of particular relevance; the claimed invention cannot be considered novel or cannot be considered to involve an inventive step when the document is taken alone

"Y" document of particular relevance; the claimed invention cannot be considered to involve an inventive step when the document is combined with one or more other such documents, such combination being obvious to a person skilled in the art

"&" document member of the same patent family

Date of the actual completion of the international search	Date of mailing of the international search report
14 December 2018	02/01/2019

Name and mailing address of the ISA/	Authorized officer
European Patent Office, P.B. 5818 Patentlaan 2 NL - 2280 HV Rijswijk Tel. (+31-70) 340-2040, Fax: (+31-70) 340-3016	Frank, Mario

INTERNATIONAL SEARCH REPORT

Information on patent family members

International application No
PCT/EP2018/075874

Patent document cited in search report	Publication date	Patent family member(s)	Publication date
US 2016328713 A1	10-11-2016	CA 2984888 A1	10-11-2016
		CN 107851111 A	27-03-2018
		EP 3292484 A1	14-03-2018
		JP 2018516030 A	14-06-2018
		US 2016328713 A1	10-11-2016
		US 2016330027 A1	10-11-2016
		US 2017302450 A1	19-10-2017
		US 2018308098 A1	25-10-2018
		WO 2016179334 A1	10-11-2016
US 2007016785 A1	18-01-2007	NONE	