Cypher Social Contracts
A Novel Protocol Specification for Cyber Physical Smart Contracts

Lars Creutz
Institute for Software Systems
Trier University of Applied Sciences
Birkenfeld, Germany
Email: l.creutz@umwelt-campus.de

Guido Dartmann
Institute for Software Systems
Trier University of Applied Sciences
Birkenfeld, Germany
Email: g.dartmann@umwelt-campus.de

***Abstract*—Social interaction is the foundation of the Internet. Nevertheless, contract drafting is a tedious task and often too complex for normal social interactions between individuals or small service providers, which has led to the rise of centralized services to manage contracts, purchases or other arrangements. In order to give regular users more privacy and a place for self-organization, this paper describes the definition of a novel protocol that allows the creation of generic smart contracts which can be processed by machines or create social interaction between humans while maintaining a high level of privacy and anonymity.**

1. Introduction

The concept of a smart contract, first defined by Nick Szabo in the 1990s [1] [2], describes the general idea of a distributed contract mechanism in the context of contract law in combination with cryptographic security, especially by using public key cryptography, for the involved parties. Most of the systems in use today, of which some are described in Section 2, require programming skills and a deep understanding of the underlying system which is challenging for non-experts. In addition, new types of security vulnerabilities arise due to misuse. As money is exchanged on these platforms, economic damage can be caused to companies and private individuals [3]. Therefore we propose a novel system, that allows to define and interact with generic contracts, which are created from reusable templates, within a distributed network, by agreeing on a secure network protocol. The proposed system can be used as a top-level layer for existing or new applications and is only responsible for processing the contract tasks, on the basis of which further actions can be performed in the respective system landscape. We predominantly focus on social aspects and want to promote self-organization within decentralized networks by making it easy to create and maintain contracts.

2. Related Works

Askemos [4] was first introduced in 2002 and describes the concept of a distributed operating system based on web technologies, especially XML. The aim of the project is to distribute information forgery-proof. For this purpose, the authors implement a virtual machine, which is defined on the basis of abstract trees and is viewed by the network participants. The users of the network vote on the correctness of the virtual machine based on their own data, which uses Byzantine Fault Tolerance in order to reach consensus. BALL [5] is the first reference implementation of the Askemos project containing some example services. However, there is hardly any literature or reports on the system.

Hawk [6] is a decentralized smart contract system whose intention is to store transactions encrypted within a blockchain in order to protect the privacy of the transactions and thus the flow of money. The authors present a compiler which translates the respective Hawk program into three parts: a public blockchain program, a user program and a manager program. The manager is trusted by the parties and can therefore see any input to the contracts without having influence on the underlying protocol. Even if they make proposals to secure the manager, it means for a contract that there must always be another party involved, which must be trusted in terms of information disclosure.

Ethereum [7] can be described as a blockchain which includes a Turing complete programming language to write decentralized applications or any sort of smart contract. The white paper describes several application areas like saving accounts, insurance and marketplaces. In general, the areas of application refer primarily to the financial sector. The execution of code that changes the status of the network is subject to a fee.

Ricardian Contracts where described in [8] as part of the Ricardo payment system. The proposed concept can be seen as guide to digitalize financial organizations. The created contracts contain machine-readable references, but are also human-readable which leads to legally binding contracts. The system took several cryptographic methods into account, for example hashing and digesting messages for integrity, encryption to keep information confidential and authentication to sign and verify interactions with the contracts.

Before we describe our protocol in more detail, we start by

explaining possible applications, the intention of the system and in how far our protocol differs from the previously described systems. Most of the used systems today are based on a blockchain and include some sort of currency within the framework. We follow a different approach, in that we only map the contract to its tasks and separate it from the payment process. Interactions inside the network never include fees, because the network does only take care of the information throughput, so that the involved parties of a contract receive the latest state to do the work, which is either computation (on an automated contract) or a real world task. Therefore we do not focus to reach an uniform level of information within the network, but instead on a social network in which honest participants check the correctness of transactions and transmit them to other nodes until the correct recipient is reached. One of the protocols design goals is to be compatible with (low-level) IoT devices in terms of easy to implement abstractions. One key research question in our work is the integration in rate limited communication networks. A typical application is a LoRaWAN-based IoT network where the capacity of the communication channel is limited [9] and the device is not directly connected to the internet. In addition to the inclusion of all types of sensors and devices, our system allows communities with a limited level of digitalization to self-organize their cultural and service offerings, thereby promoting cultural participation and local businesses without relying on external institutions. The decision not to focus on payment is based on the intended application context of the system. Research like [10] shows the risks in the technical acceptance of the cryptocurrency Bitcoin from the point of view of users and developers. Risks for users include security, third-party errors, the loss of privacy and fraud. One major challenge in the use of smart contracts for the non-technical audience lies in the potential hurdles in creating such contracts [11]. Regular service providers who want to offer their services in a small community might not be interested in a cryptocurrency, nor have the technical skills to implement a secure smart contract. Instead, the service providers can offer their services as usual, defined in a natural language, inside an easy to use system which allows them to interact with a larger audience. At the same time, the system can also be used in larger organizations, which can define and handle existing or new business processes as contracts, without revealing too much of their own system infrastructure. Some integration examples are given in Section 4 where we describe how different organizations could collaborate using our protocol and how it could be used within the IoT sector. Furthermore, we want to promote the decentralization of network infrastructures and applications while contributing to the field of secure ubiquitous computing [12] by trying to specify the protocol in a way that any type of device can be included. We are convinced that self-organization and moving away from central organizations is a significant advantage in terms of data protection and privacy. Furthermore, an open protocol, which can also be used on inexpensive hardware, offers advantages in many different application areas. In summary, we see our system as a link between smart contracts and Ricardian contracts [8]. There is a kind of legal binding, as the contracts can be written in human-readable language. However, the specification of the protocol allows an automation of the actions to be performed. Our focus is not on finance, but on facilitating interactions between individuals or devices in the form of simple contracts.

3. System design

3.1. Motivation

Our idea of the system knows some similarities to the Cypherpunk's Manifesto by Eric Hughes [13] and therefore refers to the same in the following specification and in the title of this paper. The goals and problems stated in 1993 are still valid today. His work addresses the need for freedom of speech for every party inside a forum of an open society. Communication within the network must therefore be secure for the individual parties, yet transparent for all other participants, in order to implement a forum of this kind and to consolidate a widely distributed network. The information disclosed must be kept to a minimum to protect individual privacy and must be correspondingly securely encrypted. Furthermore, the system must be open and distributed, so that everyone is given the opportunity to participate. In addition, no central unit should be able to force how the system may be used, allowing the development of abstractions to further enhance the spread and use of the system. Hughes describes the difference between privacy and secrecy[1], which should be clarified in the context of our protocol. Every network participant must have the right to disclose only the information he wants and still make his contribution to the functioning of the network. In order to interact with a particular contract, it may be necessary to disclose private information. The user must therefore be able to provide this information to the other party in a secure manner. Providing private information always carries the risk of interest tracking and personal profiling. Within our system there must be the possibility to change the identity continuously, so that not only a minimum of data is securely exchanged between two parties, but also that the identity does not have to be used again afterwards. To reach privacy and secrecy, we rely on several cryptographic procedures which we describe within the next sections. The user is secret in terms of the interaction with the network because only the current IP address[2], without any direct reference to the origin of the transaction, is known from the point of view of the other users.

1. "A private matter is something one doesn't want the whole world to know, but a secret matter is something one doesn't want anybody to know." [13]

2. We note that this does not mean, that a user can't be tracked by using geo-localisation. It just means that a party can't be sure if the received transaction was created or just forwarded by that IP address. Nevertheless, any user could be behind a VPN or proxy to increase his anonymity.

3.2. Protocol definitions

Account. An account $\mathcal{A}_i$ of a user i is defined as a tuple of private (secret) $\mathcal{K}_{i,S}$ and public key $\mathcal{K}_{i,P}$ in the field of public key cryptography.

$$\mathcal{A}_i = (\mathcal{K}_{i,S}, \mathcal{K}_{i,P}) \tag{1}$$

The public part $\mathcal{K}_{i,P}$ of the account is included in all transactions so that every network participant has the possibility to verify the authenticity of the received message, since it must be signed using $\mathcal{K}_{i,S}$.

Task. We define a task t as the hash value of its description $\mathcal{D}$:

$$t_i = \mathcal{H}(\mathcal{D}_i) \tag{2}$$

If the confirmation of a task t_i requires that data d has to be transmitted to the other party j, we use ephemeral key pairs $\mathcal{K}_{\mathcal{E}}$ to encrypt the data and reach forward secrecy [14]. Each party embeddes a new public key during in the contract creation process. To encrypt data during the actual processing of the tasks, the temporary keys are combined to generate a shared secret, which is used to symmetrically encrypt the input $\mathcal{I}(d)$ to the contract $\mathcal{C}_{i,j}$ for the respective task t_i.

$$i \rightarrow j : \{\mathcal{I}(d)\}_{(\mathcal{K}_{\mathcal{E}\,j,P}, \mathcal{K}_{\mathcal{E}\,i,S})} \tag{3}$$

Hence only the correct recipient of the transaction is able to determine the specific content of the message by decrypting it with the same shared secret. Additionally the ephemeral keys can be destroyed after a contract was finished. Leaking the private key would comprise the account but it still would not be possible to access the contents of fulfilled contracts. At the same time, all other network participants confirm the integrity of the encrypted data while validating the signature.

Template. The template is the base of which a contract is derived from and must be published prior to the contract. Our approach allows to reuse the same template for an unlimited amount of contracts. The reusability of the task definitions, in combination with the chosen data structure, allows to automate processes.

$$\mathcal{O}_j = (\mathcal{H}_{\mathcal{O}_j}, \mathcal{K}_{j,P}, r_M, T, R, B) \tag{4}$$

The template is defined by the receiver of the contract j and contains the task definitions $\mathcal{D}$ in $T = \{\mathcal{D}_1, \mathcal{D}_2, .., \mathcal{D}_N\}$ which are used to calculate a Merkle tree [15] and its root hash r_M. A Merkle tree $\mathcal{M}$ is defined [16] as a binary tree $\mathcal{M} = (\mathcal{V}_M, \mathcal{E}_M)$ with a set of nodes $\mathcal{V}_M$ and a set of edges $\mathcal{E}_M$ and a root $r_M \in \mathcal{V}_M$. Merkle trees are used to keep the amount of data to be transmitted within the network as low as possible and create the functionality to easily look up tasks and map them to a specific action in the respective systems. New contracts do not need to include all task definitions, because they are given in the template and are referenced by r_M. The defined tasks must be completed in the correct order for the network to determine the actual state of a contract. For this purpose, $R = \{r_1, r_2, ..., r_N\}$, maps the affiliation of a task to a party. Network participants also use the Merkle tree to check if the correct task was addressed. Besides the definitions of the tasks, the template also includes a description B. The respective templates are not semantically checked, so the description and/or task definitions can be completely anonymous/obfuscated and the actual content of the contract can be agreed upon via other channels. Any published template is in the state *active* and can be revoked by the creator, changing the state to *inactive*. After a revocation, a template cannot be reactivated.

TABLE 1. OVERVIEW OF A CONTRACTS STATE

State	Definition
init	The contract has been published (offer) and references to a given contract template
rejected	The creator of the contract template rejected the offer
live	The receiver (publisher of the contract template) accepted the contract
finished	All tasks of the contract have been completed

Contract. Contracts consist of two parties i and j which have agreed on a set of tasks using a the publicly available template.

$$\mathcal{C}_{i,j} = (\mathcal{H}_{\mathcal{C}_{i,j}}, \mathcal{K}_{i,P}, \mathcal{K}_{j,P}, N, r_M, \mathcal{O}_j, R) \tag{5}$$

Every contract contains a reference to an active template $\mathcal{O}_j$ and reuses several elements of that template to increase the performance when validating transactions during the contract execution, because network participants only need to access the template when adding the contract to their local state.
Table 1 shows the states of a contract within our protocol.

Transaction. A transaction $\mathcal{T}_i$ of a user i is defined as:

$$\mathcal{T}_i = (\mathcal{H}_{T_i}, \mathcal{S}_i, \mathcal{K}_{i,P}, data, type, \mathcal{K}_{\mathcal{E}\,i,P}{}^{3}) \tag{6}$$

where $\mathcal{H}_{T_i}$ is the unique hash of the transaction, $\mathcal{S}_i$ the signature by user i of $\mathcal{H}_{T_i}$ using $\mathcal{K}_{i,S}$, $\mathcal{K}_{i,P}$ the public portion of the users account $\mathcal{A}_i$ and data/type define the nature of the transaction corresponding to Table 2.

3.3. Template/Contract creation

To illustrate the protocol definitions we introduce a simple example that is referenced in the next sections. We assume a service provider j that offers a specific good including a delivery and begin with the template definition. Based on (4) the template $\mathcal{O}_j$ includes $T = \{$*"Specify amount and delivery address", "Deliver amount of product to the address", "Confirm the delivery", "Confirm the payment"*$\}$ and $R = \{S, R, S, R\}$. After the definition of

3. The ephemeral public key $\mathcal{K}_{\mathcal{E}\,i,P}$ is only added if the type of a transaction is either an offer or the acceptance of a contract

TABLE 2. DATA AND TYPE OF A TRANSACTION DEPENDING ON ITS APPLICATION CONTEXT

Type	Data
Confirm Task	Contract hash, task, proof, transaction reference, input data
Contract Offer	Entire contract
Contract Accept	Hash of the accepted contract
Contract Decline	Hash of the declined contract
Template Publish	Entire template
Template Revoke	Hash of the active template

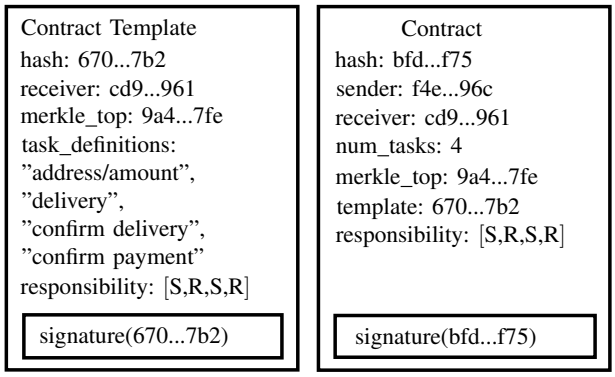

Figure 1. Overview of the resulting template and contract. The figure is simplified and is intended for clarification only. Therefore the keys are shown as fingerprints and not in the real encoding format. Likewise the signatures shown also refer to the hash of the template/contract, although they refer to the respective transaction in the implementation.

$\mathcal{O}_j$ the corresponding transaction (6) $\mathcal{T}_j$ is created where $data = \mathcal{O}_j$ and *type* indicates that the sender of the transaction wants to publish a new template. The customer i accesses the publicly available, active contract template $\mathcal{O}_j$ to generate a new contract $\mathcal{C}_{i,j}$ and publishes it within the network. After the transaction of user i with (6) where $type$ describes an offer and $data = \mathcal{C}_{i,j}$ has reached the service provider j, the latter can decide whether to accept the contract and thus start the processing. Fig. 1 shows the relationship between template and contract within our system design. The generated hash values of the template and contract are based on the concatenation of the given input values. In addition to the contents shown, the protocol contains further implementation-specific fields such as timestamps and nonce values to guarantee that the generated hash values are unique. Furthermore, templates contain a description, which will make it possible to search for a specific contract template and implement an easy to use marketplace in the future. To be compatible and open for abstractions, the nonce value of a contract can be specified by the user. It is important to note that a reused nonce will lead other nodes to decline the contract because it is already present in their local state. Nevertheless we need to be able to generate a contract with as low input data as possible. When using an active template in our LoRaWAN use case, we can generate a nonce for the contract on the low-level device, which will be reused by the abstraction to recreate the same contract. Any interaction between the two components would refer to the same contract, even though it was never transmitted via LoRaWAN due to the rate limits. The associated transaction is only known at the abstraction and never seen by the device. We note that the framework to be released in the future may slightly differ from the presentation, as implementations are evaluated and improved to provide a stable and secure reference implementation.

3.4. Contract interaction

In order to interact with the contract $\mathcal{C}_{i,j}$, a transaction $\mathcal{T}_i$ must be transmitted to the network which must reach the other party j either directly or indirectly. Participants who receive the transaction must first check whether the signature $\mathcal{S}_i$ contained in the transaction is valid by using the corresponding public key $\mathcal{K}_{i,P}$. Given a valid signature, the nodes also check whether the received transaction $\mathcal{T}_i$ matches their local state of the contract $\mathcal{C}_{i,j}$. If the transaction $\mathcal{T}_i$ refers to a state that is unknown to the participant (for example: a different number of tasks already performed), other network participants are instructed to forward the previous transactions. This can lead to a chain of requests, which propagates through the network until the desired information is available. In order to simplify this procedure and at the same time ensure that the respective participants have specific information about the transaction, each transaction $\mathcal{T}_i$ refers to the previous transaction $\mathcal{T}_l$ i.e. $\mathcal{T}_i \rightarrow \mathcal{T}_l$ by including the hash $\mathcal{H}_{\mathcal{T}_l}$. This facilitates the search process within the network to determine whether the contract is processed in the correct order. To ensure the integrity of the protocol, the transactions also contain a proof (authentication path [15]) which the other nodes use to calculate the respective Merkle top (root) hash r_M. We note that checking integrity cannot tell whether the task has actually been performed or not. However, for the network participants who only receive the transaction and are not part of the contract, only the reference to the correct previous transaction and the proof that the task addressed belongs to the contract is important. This procedure allows other network participants to validate a transaction $\mathcal{T}_i$ by knowing only the original contract $\mathcal{C}_{i,j}$, without accessing the associated template that was used to create it, after the contract was accepted. The presence of the number of tasks N in the actual contract also prevents second preimage attacks [17], as it must contain a proof related to the number of tasks N. Given the previous example, every network participant who knows the contract $\mathcal{C}_{i,j}$ now knows that the number of tasks is $N = 4$. Thus the structure of the tree is determined. The respective proofs are stored as arrays, whose fields refer to the nodes within the tree. Each transaction must send $\log_2(N)$ additional hash values to verify the integrity of the transaction [15]. Fig. 2 shows an example tree with 4 tasks. In general, the protocol

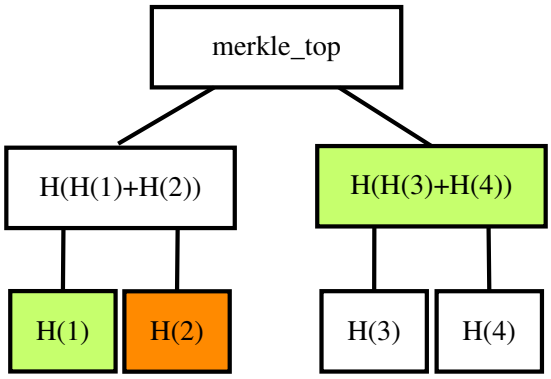

Figure 2. Merkle tree generated from the previously introduced example. The orange node refers to the task which the transaction wants to confirm. The green nodes are necessary for other network participants to check the integrity

guarantees for the number of tasks:

$$\log_2(N) \in \mathbb{N} \tag{7}$$

by adding padding (empty tasks) to the end of the list if needed. Before we show the proof for the second task within the contract, we will explain the transmission of the first task confirmation for the sake of completeness. Since the first task cannot refer to any previous one, it refers to the transaction of the template creator who accepted the offer for the contract and thus set the status from *init* to *live*. To describe the confirmation we use the following more readable notatation:

$$\mathcal{H}_i = H(i) \tag{8}$$

$$\mathcal{H}_{i,j} = H(H(i) + H(j)) \tag{9}$$

The transaction $\mathcal{T}_j$[4] to confirm the (current) second task $t_{c=2}$ needs to include the values of $\mathcal{H}_1$, $\mathcal{H}_{3,4}$ and $\mathcal{H}_2$, resulting in a data structure similar to:

$$\mathbf{p} = [0, 0, \mathcal{H}_{3,4}, \mathcal{H}_1, \mathcal{H}_2, 0, 0] \tag{10}$$

The index of $\mathcal{H}_2$ must match the following condition (with the consideration of possible padding):

$$index = c + N - 2 \tag{11}$$

The other network participants use those hash values to calculate the top hash of the tree r_M and compare it to the value which was used to define the contract $\mathcal{C}_{i,j}$. Because a full binary tree is created by (7), it is sufficient for the recipients of the message to perform the following operations to validate a transaction $\mathcal{T}_j$ with (6) where $data = \{\mathcal{H}_{\mathcal{C}_{i,j}}, t_2, \mathbf{p}, \mathcal{T}_l\}$ and $type$ indicates a confirmation:

1) Test if the signature $\mathcal{S}_j$ of the transaction $\mathcal{T}_j$ is valid and matches the known public key $\mathcal{K}_{j,P}$ published in the contract $\mathcal{C}_{i,j}$
2) Confirm that the task t_2 addressed by the transaction is valid for the party that issued the transaction
3) Get the previous transaction $\mathcal{T}_l$ from the transactions sender or other nodes if necessary
4) Build the tree by the given values in $\mathbf{p}$
5) Check if the calculated top hash is equal to the contracts top hash r_M
6) Update the local state of the contract and broadcast the transaction to other users

Each of the nodes in between the process of the transportation of the transaction also check the validity of the transaction by applying the described method. The protocol defines that a trusted peer does not redirect an untrusted transaction, which contributes to the social intention principle and will exclude non-working peers that only try to mirror and redirect transactions without validating the signatures. Since the transactions can be understood as a receipt and the uninvolved network participants who may receive, validate and transmit the transaction cannot necessarily be trusted to archive the transaction indefinitely, the parties included in the contract must store the transactions in order to document the current status of the contract.
In general, the transaction addresses its destination by referring to a unique point of the network, which can be a contract or a template. An abstraction to make the protocol available for an unsupported device must only ensure secure communication between itself and the particular device, including a scheme to address a contract or template.

3.5. Network interaction and architecture

In the following we provide an insight into the network architecture of the overall system. In general, we implement a peer to peer network to support the core functionality of the described system. The further development of network abstractions for using our system with low-level IoT devices is not described in this paper. Challenges in the structure and implementation of peer to peer networks are described in [18] and [19]. The protocol provides that each participant is able to determine his communication partners, as well as the number of connections maintained, which provides the possibility of running private subnetworks of the protocol. We distinguish between full nodes, which accept incoming connections, and normal nodes that connect to full nodes to publish their transactions. Normal nodes will still receive other transactions, broadcasted by full nodes, and can apply filters to decide whether the transaction should be included in their own database. The type of node is exchanged on the first interaction between two nodes. If a node states to be a full node, it is added to a local database that manages the available endpoints. Depending on the specified number of active connections, each connection establishes additional connections to other nodes, which are used for broadcasting and when the original connection is interrupted. Those additional connections are shared inside a factory object, so that the main connections do not connect to the same backup nodes. Within the main scope of the frameworks'

4. In our example, the receiver j is responsible for the second task, which is illustrated in Fig. 1

TABLE 3. SUBSET OF THE MESSAGES INSIDE THE PEER TO PEER NETWORK WITH CORRESPONDING RESPONSES

Message	Description	Response
PING	Keep alive of a connection and feedback to other nodes	PONG
DISCOVER	Ask for a nodes other peers	List of peers running full nodes
CONTRACT (GET)	Get the specific contract identified by its hash	Requested contract if present
TEMPLATE (GET)	Receive information about the requested template	Related template to create contracts with
TRANSACTION (GET)	Request a specific transaction	Corresponding transaction if present
STATE (GET)	Information about the current state of an object (contract or template)	Point of view of the object with the local information
PUBLISH	Publish a new transaction	Confirmation that the node accepted the transaction
INFO	Information about the current type of a node	Node information of the receiving peer

usage, new full nodes are added automatically, whereas in private networks all available endpoints must be specified by the user.

In Table 3 we describe some messages that are exchanged between the participants. These messages do not refer to the transmission of the actual contracts or templates, but to the implementation of the networking protocol. The list is not exhaustive and only clarifies how communication takes place and how the respective participants adjust their level of knowledge. The network interaction should be kept to a minimum. It must be possible for the participants to keep an overview of who is currently contributing to the network by continuously pinging the other nodes. The option to disclose active connections of other users allows, in case of a fluctuating number of communication partners, to establish further connections to previously unknown network participants, in order to maintain a certain number of active connections. In addition, it must be possible for nodes to efficiently exchange information about the current state of the network. For this purpose, each participant checks, after the connection to the network has been established, whether new transactions are available, based on the timestamp of the last received one. If the local status differs, the respective existing transactions are exchanged. The most important aspects for participating in the network are an internet connection for receiving and forwarding messages and sufficient storage space, which is becoming increasingly cheap.

3.6. Reaching consensus

We define consensus as the correct level of information for every party involved in the interaction. Our system can be seen more as a forum and does not represent a distributed ledger. Therefore consensus is based on the resulting surplus value for the users of the system and the number of honest network participants. Every honest contributor must record all necessary transactions in order to document the state of a contract, as the protocol requires that the transactions are carried out in the correct order. For this purpose, it is sufficient to store the signed transactions, which refer to the respective contract or template, and the local state of the network which is built from the received transactions. Since our goal is to provide the basic system and leave it openly configurable for many purposes, we expect that gradually central instances will emerge, which will be trusted by some parts of the users. An example would be a trusted third party that provides a contract template for obtaining information about service providers. If a service provider openly discloses his identity there, the central authority, which must be trusted by the user in this case, can make a statement about the reputation of the service provider and in this case increase the probability that the respective contract will be processed accordingly. Assuming the application context for automated processing of contracts in the form of devices, it can also be expected that those contracts will be processed as intended. The system therefore only supports their cooperation and administration by providing an abstract interaction layer and a distributed data management for the users that follow the protocol. We still want to point out a possible attack scenario when trying to cheat inside the network. A dishonest user could send two different transaction referring to the same task to various nodes. Hereby the malicious user would try to get surrounding nodes to accept both states. If that intentionally wrong transaction reaches the other party of the contract and leads to the next step being fulfilled with wrong input to the previous task, the whole contract would be compromised. Nevertheless, the other party has the proof of a signed transaction by the malicious user, because he included it inside the next task. Honest nodes would be able to identify the dishonest network participant quickly. Furthermore, there is zero to no benefit on trying to cheat because we focus on the abstract tasks themselves and do not need to have a single true state of the network. All honestly performed tasks, which are based on dishonest previous transactions, can be clearly clarified within a legal dispute. We believe that there are enough secure digital currencies in the currently available systems and their derivatives.

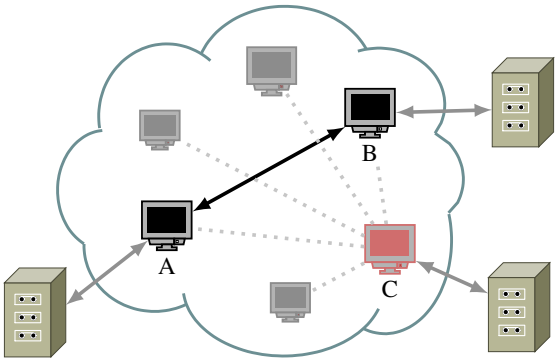

Figure 3. Interaction between two parties that is observed by a third party involved. The actual systems of the parties are not part of the public network, which contributes to system security and simplifies cooperation

Therefore, our protocol focuses only on the decentralized creation of contracts in combination with the validation of the interactions between the network participants. We open a digital forum for self-organization, which can be used by everyone for any purpose. The payment process can take place in any form and depends only on the agreement reached between buyer and seller.

4. System integration

To conclude the description of the system, we give some examples of the wide-ranging integration possibilities and application areas of the protocol in the following. As mentioned, we see the system as an opportunity for structurally weak regions in terms of digitalization to promote self-organization. Services can be offered and processed conveniently inside a decentralized platform over which the users retain full control. Likewise, no additional systems are required for interaction, since a peer to peer network eliminates the need for a central instance. Additionally we emphasize the new possibilities for completely isolated areas to use similar services and to process the contracts accordingly via the mentioned LoRaWAN abstraction. Overall, social interaction can benefit from our system in general, as no expert knowledge will be required to use the system safely, allowing not only service providers but also regular agreements to be documented inside an open network. The data can be stored distributed on various devices, that are continuously aware of the current state of the network, reducing the vulnerability of a central system. In addition to the social interaction of private individuals, the system can also be interesting for business cooperations of many kinds, as it allows the respective individual systems to be encapsulated from cooperation partners while still being able to collaborate. The transparent interactions also enable a far-reaching knowledge alignment, on the basis of which individual activities are defined. Fig. 3 shows a direct cooperation between two contract partners. In our example, we assume that party A is currently cooperating with party B, while party C is not directly involved in the contract. However, if party C knows the context of the interaction between A and B, and possibly collaborates with party A in the future, actions can be performed in party C's system that improve the subsequent workflow with party A without connecting the underlying systems of the three parties. The network shown in Fig. 3 could at the same time be entirely private, allowing the parties to agree on who is able to monitor their interactions, increasing the likelihood to even consider to use the protocol inside a closed company infrastructure. Additionally the created contracts could be used to control ubiquitous devices. One simple example would be the control of devices within a smart home environment. Imagine a device that handles the temperature control and manages the corresponding contract template. Within the automation process it can be specified that only contracts of known accounts are accepted. During contract processing, the contract creator now specifies the desired temperature as data input to the confirmation of the respective task, which is then set by the device. Possible attack scenarios and concerns regarding privacy, described in [20], can be prevented while having full control over your device and your personal data. Finally, we would like to emphasize the automatability of the respective processes within contract processing. Recurring tasks, which are defined in the same way in different contracts, are clearly identifiable by their hash value $\mathcal{H}$. The use of a Merkle tree therefore not only allows the automation of the respective tasks, but also creates a context between task and contract. Thus a logical separation and classification can be made within the automated processing. One simple example would be the distinction between private and business customers inside an order process. The task for transmitting contact data (address, mail, etc.) can be defined in the same way and can trigger the respective action in the target system, whereas the contract provides the context, such as how the invoice is created with regard to the respective taxes.

5. Conclusion and future work

We have presented a novel protocol for the creation and execution of *Cypher Social Contracts* which presents a different view on the field of smart contracts. We focus on the anonymity of the users and on ensuring that the correct state of the contract processing cannot be distorted as long as there are enough network participants distributing the transactions. The information to be provided varies depending on the content of the contract and thus minimizes the necessity to disclose private information. Likewise, by using transparent contract templates, users can see exactly what information they need to disclose. The system deliberately protects its users and contributes to data protection in an era where personalized advertising is omnipresent. In addition, our approach enables the use of the protocol in private networks without the need to make the application area or the underlying infrastructure publicly known. This is not only beneficial for collaborations between companies, but can also be used in the private sector to establish contracts

between devices that execute tasks automatically within an encapsulated infrastructure.

As already mentioned in the paper, our current main focus is on providing a compatible LoRaWAN abstraction to use the postulated system and express its benefits for low-level devices in the IoT area. Therefore we are currently implementing a secure middle man service to be able to interact with a contract without any sort of internet connectivity on the devices that provide the input data. Our primarily focus lies on mobility on demand applications and on advancing the generic smart contract platform for any other domain to use it. Furthermore, a marketplace will be built and tested for those application domains where sensors, persons and cars will interact with each other. Lastly, we test and extend the framework and its abstractions in order to make them publicly available soon.

Acknowledgment

This work has been funded by the Federal Ministry of Transport and Digital Infrastructure (BMVI) within the funding guideline "Automated and Connected Driving" under the grant number 16AVF2134C and under the research initiative "mFUND" under the grant number 19F2102F.

References

[1] N. Szabo, "Smart contracts," http://www.fon.hum.uva.nl/rob/Courses/InformationInSpeech/CDROM/Literature/LOTwinterschool2006/szabo.best.vwh.net/smart.contracts.html, 1994, [Online - accessed 04-15-2020].

[2] ——, "Smart Contracts: Building Blocks for Digital Markets," http://www.fon.hum.uva.nl/rob/Courses/InformationInSpeech/CDROM/Literature/LOTwinterschool2006/szabo.best.vwh.net/smart.contracts_2.html, 1996, [Online - accessed 04-15-2020].

[3] A. Mense and M. Flatscher, "Security Vulnerabilities in Ethereum Smart Contracts," in *Proceedings of the 20th International Conference on Information Integration and Web-Based Applications & Services*, ser. iiWAS2018. New York, NY, USA: Association for Computing Machinery, 2018, pp. 375-380. [Online]. Available: https://doi.org/10.1145/3282373.3282419

[4] F. Wittenberger, "Askemos-a distributed settlement," 2002.

[5] ——, "BALL - Askemos," http://ball.askemos.org/, [Online - accessed 04-20-2020].

[6] A. Kosba, A. Miller, E. Shi, Z. Wen, and C. Papamanthou, "Hawk: The Blockchain Model of Cryptography and Privacy-Preserving Smart Contracts," in *2016 IEEE Symposium on Security and Privacy (SP)*, 2016, pp. 839–858.

[7] V. Buterin *et al.*, "Ethereum white paper," https://github.com/ethereum/wiki/wiki/White-Paper, 2013.

[8] I. Grigg, "Financial Cryptography in 7 Layers," in *Financial Cryptography*, Y. Frankel, Ed. Berlin, Heidelberg: Springer Berlin Heidelberg, 2001, pp. 332–348.

[9] F. Adelantado, X. Vilajosana, P. Tuset-Peiro, B. Martinez, J. Melia-Segui, and T. Watteyne, "Understanding the Limits of LoRaWAN," *IEEE Communications Magazine*, vol. 55, no. 9, pp. 34–40, 2017.

[10] D. Folkinshteyn and M. Lennon, "Braving Bitcoin: A technology acceptance model (TAM) analysis," *Journal of Information Technology Case and Application Research*, vol. 184, pp. 220–249, 03 2017.

[11] Stuart D. Levi and Alex B. Lipton, "An Introduction to Smart Contracts and Their Potential and Inherent Limitations," https://corpgov.law.harvard.edu/2018/05/26/an-introduction-to-smart-contracts-and-their-potential-and-inherent-limitations/, [Online - accessed 22-09-2020].

[12] M. Weiser, "The Computer for the 21st Century," *SIGMOBILE Mob. Comput. Commun. Rev.*, vol. 3, no. 3, p. 311, Jul. 1999. [Online]. Available: https://doi.org/10.1145/329124.329126

[13] E. Hughes, "A Cypherpunk's Manifesto," http://www.activism.net/cypherpunk/manifesto.html, 1993, [Online - accessed 04-25-2020].

[14] W. Diffie, P. C. V. Oorschot, and M. J. Wiener, "Authentication and Authenticated Key Exchanges," 1992.

[15] R. Merkle, "Protocols for Public Key Cryptosystems," 04 1980, pp. 122–134.

[16] M. Ogawa, E. Horita, and S. Ono, "Proving Properties of Incremental Merkle Trees," in *Automated Deduction – CADE-20*, R. Nieuwenhuis, Ed. Berlin, Heidelberg: Springer Berlin Heidelberg, 2005, pp. 424–440.

[17] B. Laurie, A. Langley, and E. Kasper, "Certificate Transparency," https://www.rfc-editor.org/info/rfc6962, 2013, rFC 6962, DOI 10.17487/RFC6962.

[18] D. Bandara and A. Jayasumana, "Collaborative Applications over Peer-to-Peer Systems - Challenges and Solutions," *Peer-to-Peer Networking and Applications*, vol. 6, 07 2012.

[19] W. Nejdl, W. Siberski, and M. Sintek, "Design Issues and Challenges for RDF- and Schema-Based Peer-to-Peer Systems," *SIGMOD Rec.*, vol. 32, no. 3, p. 4146, Sep. 2003. [Online]. Available: https://doi.org/10.1145/945721.945731

[20] A. Dasgupta, A. Q. Gill, and F. Hussain, "Privacy of IoT-enabled smart home systems," in *Internet of Things (IoT) for Automated and Smart Applications*. IntechOpen, 2019.