Towards Secure Distributed Trust Management on a Global Scale

An analytical approach for applying Distributed Ledgers for authorization in the IoT

Nikolaos Alexopoulos
Technische Universität Darmstadt
Darmstadt, Germany
alexopoulos@tk.tu-darmstadt.de

Sheikh Mahbub Habib
Technische Universität Darmstadt
Darmstadt, Germany
sheikh@tk.tu-darmstadt.de

Max Mühlhäuser
Technische Universität Darmstadt
Darmstadt, Germany
max@informatik.tu-darmstadt.de

ABSTRACT

Authorization, and more generally *Trust Management (TM)*, is an indispensable part of the correct operation of most IT systems. The advent of the Internet of Things (IoT), with its cyber-physical and distributed nature, creates new challenges, that existing TM systems cannot adequately address, such as for example the need for non-interactive exclusive access enforcement. In the meantime, a line of thought in the research community is that *Distributed Ledgers* (DLs), like the one implemented by the Ethereum blockchain, can provide strong security guarantees for distributed access control. However, this approach has not yet been examined in a scientific, systematic manner, and has many pitfalls, with arguably the most important one being scalability.

In this paper, we critically explore the shortcomings of existing solutions for trust management in distributed networks, pinpoint which of these shortcomings can be addressed by utilizing DLs, and offer a conceptual design for a scalable, secure TM system. Our design approaches the problem in three layers, namely a *global*, an intermediate *group* or *shard* layer, and a *local* layer, corresponding to the set of embedded devices behind an internet access point. We view our design as a novel first step, helping the community to produce more secure and realistic authorization solutions for the IoT, in the near future.

CCS CONCEPTS

• **Security and privacy** → **Access control**; *Distributed systems security*; • **Computer systems organization** → *Peer-to-peer architectures*;

ACM Reference Format:
Nikolaos Alexopoulos, Sheikh Mahbub Habib, and Max Mühlhäuser. 2018. Towards Secure Distributed Trust Management on a Global Scale: An analytical approach for applying Distributed Ledgers for authorization in the IoT. In *IoT S&P'18: ACM SIGCOMM 2018 Workshop on IoT Security and Privacy , August 20, 2018, Budapest, Hungary*. ACM, New York, NY, USA, 6 pages. https://doi.org/10.1145/3229565.3229569

IoT S&P'18, August 20, 2018, Budapest, Hungary

ACM ISBN 978-1-4503-5905-4/18/08…$15.00
https://doi.org/10.1145/3229565.3229569

1 INTRODUCTION

Authorization or *access control*, is a traditionally indispensable part of security, whether we think of the physical or the digital world. In the physical world, access control is facilitated through various means; for instance with physical keys (e.g. for apartments, cars, etc.), paper credentials (e.g. passports for boarding a flight), or forms of passwords (e.g. a password to open a safe). Decisions on whether or not to delegate access to another individual are most often taken based on *trust* derived from personal interaction and social/legal structures. For example, the owner of a house, i.e. the person whose name is on the contract, would issue credentials (physical keys) to members of her family, or, for a limited time, to a guest who is spending time in the house.

In the digital world, *authorization* is traditionally associated with permission rights on files residing in a system (either local or remote). The most characteristic example is the UNIX family of operating systems [1] , which offers a variety of access control methods, like mandatory and discretionary access control (MAC and DAC respectively) [25], access control lists (ACLs), as well as more advanced methods like role-based access control (RBAC) [24]. In a remote scenario, authorization is most commonly achieved through centralized services, like X.509 certificates [12] and Kerberos [21]. The need for distributed authentication and authorization gave birth to the notion of *Trust Management* [4, 5], which paved the way for distributed, generalized and abstract trust inference systems that can implement policies adaptively, according to the intentions of the party holding the resource under question.

The emergence of distributed networks of embedded devices (coined as the "Internet of Things" (IoT)) that can offer digital, as well as physical services, generates new challenges for trust management. Traditional schemes, even decentralized, as the ones introduced above, suffer from several shortcomings: (a) they cannot implement complex policies in non-interactive (IoT) environments (e.g. exclusive access delegation - `ONLY U1 CAN ACCESS D1`, elective access delegation - `ACCESS ONLY 1 OF {D1,D2,D3}`) (b) they do not incorporate *incentives* for good behaviour, (c) they do not take past experiences into account when making decisions (whether or not to grant access or issue delegation credentials), (d) they cannot guarantee the transparency of "negative" trust information (e.g. revocations, negative experiences), thus they cannot implement "negative" rules (e.g. `ALLOW IF A AND (NOT B)`), (e) they are not designed for embedded devices, and thus do not consider *context* in rule specification (e.g. `ALLOW IF DEVICE A IS NEAR DEVICE B`), and (f) they cannot implement time-limited rules, (e.g. `ALLOW IF T1<t<T2`), or they assume some kind of global clock.

[1] For a more complete view of authorization in UNIX, we refer the reader to [9].

To address the shortcomings of current trust management systems, new, more powerful techniques need to be developed and applied. On a higher level, the ultimate goal would be to *simulate* a trusted third party (TTP) through a distributed mechanism, and therefore be able to enforce arbitrary policies, rules, penalties, etc. Distributed ledgers (DLs), such as the ones implemented by blockchain technology (see section 2.1) intuitively fit our scenario, as they can simulate a TTP (under well-defined security assumptions) by using consensus algorithms. Specifically, decentralized platforms such as *Ethereum* [29], that can run arbitrary (Turing-complete) programs (smart contracts) in a secure manner, can be used to address various of the shortcomings presented above, and under well-defined assumptions guarantee provably secure properties. Additionally, due to their use in cryptocurrency implementations, they can seamlessly authorization delegations with related financial transactions (e.g. pay for using a car, an internet access point or reading data from a sensor). However, due to the overhead incurred by the operation of DLs, it is of vital importance to recognize exactly for what operations, reliance on a DL is necessary. Moreover, the scalability of the mechanisms is, arguably, the most serious limitation of the design space, as the number of embedded devices that would potentially be handled by the system is huge. Therefore, the system should be "scale-out", meaning that the introduction of new parties/devices in the system should have negligible effect on the complexity of each operation.

Contributions:
In this paper, we critically explore the design of a distributed trust management system that can scale to global dimensions. We begin by presenting a straw-man design that can be used to study the advantages of using DLs as a part of a trust management system, as well as critically discern scenarios when DLs do not offer sizeable security benefits, and may even incur negative consequences like time delays and transaction fees. Then, towards scalable and secure trust management on a global scale, we propose a *hybrid* design, following a 3-layer architecture. By *hybrid*, we mean that access delegations and trust assessments can be exchanged both in a simple peer-to-peer (P2P) way, as well as through a DL. To achieve scalability and reduce unnecessary overhead, we dissect the problem into 3 layers:

- **(Layer2) Global:** Security of operations among members of different clusters/shards of Layer1 (see next point) is guaranteed by using a global public DL, such as *Ethereum*, and encoding access control logic in it.
- **(Layer1) Shards:** Shards are groups of internet access points, which interact often with each other. Can be thought of as the "city" level. Traditional Byzantine fault tolerant algorithms [7] can be used to construct a DL, as e.g. implemented by *Hyperledger Fabric*[6].
- **(Layer0) Local:** Embedded (or not) devices and users behind a single internet access point (router). Access points can be considered honest (trusted), semi-honest (privacy threat), or malicious (actively attempts to manipulate access decisions). They have access to both upper layers.

Finally, we articulate the challenges faced at each layer that the community should overcome in order to produce *practical* and *secure* trust management systems for cyber-physical environments.

2 BACKGROUND AND RELATED WORK

In this section we present a brief overview of distributed ledgers and a summary of the related work in the field of trust management.

2.1 Distributed ledgers and smart contracts

A Distributed Ledger (DL) can be thought of as a data file (a ledger - book) that is replicated among a number of peers. The contents of the file and their modifications are agreed upon by the peers via consensus algorithms. Although the problem of consensus in open networks (unknown number of participants) is known to be impossible to solve, blockchain technology, as introduced by Bitcoin [20] in 2008, provided a breakthrough by using proof-of-work (racing for the solution of a difficult puzzle) to achieve (probabilistic) consensus even in open networks. More recently, proof-of-stake designs (e.g. [11, 16]) have paved the way for consensus in open networks without the necessity of energy waste as a result of proof-of-work.

At first, the ability to achieve consensus in open networks was utilized to create decentralized digital currencies, like Bitcoin and the numerous alt-coins that exist. However, *Ethereum* [29] introduced the possibility of storing Turing-complete code (smart contracts) on the ledger, that can change the state of other variables stored on it, and the correct execution of which is guaranteed based on the ledger's consensus properties. Peers can deploy or call functions stored on the DL and change the state of the DL in a consistent manner (maintaining consensus), i.e. all honest participants have the same view of a certain snapshot of the ledger (the state at time t) with overwhelming probability as time passes.

In public distributed ledgers, like Bitcoin and Ethereum, nodes who participate in the consensus algorithm are called *miners*, and are the ones guaranteeing the consensus properties of the ledger. In order to incentivize miners to invest computational resources, cryptocurrency fees need to be paid by parties who want to modify the ledger. In Ethereum, these fees are called *gas*, and they depend on the computational and space complexity of the smart contract to be executed. Other nodes running the full version of the protocol, but which do not participate in the consensus process, are known as *full* validating nodes. Although they don't have the ability to make changes to the ledger, they can monitor it and validate that the changes made are in accordance with the rules of the protocol. Nodes who cannot (due to limited computational or network resources), or do not want to inspect and validate every change that occurs in the ledger, but want to know specific parts of the state, are known as *light clients*. The security of light clients, i.e. their assurance that the state they observe is indeed the global state, is weaker than the full security guarantees of full validating nodes. However, efficient algorithms exist (e.g. [27]) that offer them substantial security guarantees with a need for limited communication with full validating nodes.

The developments in achieving consensus in open/public networks rekindled the interest of the community on achieving consensus in permissioned networks (peers known and authenticated) by using Byzantine fault tolerant algorithms, e.g. [7]. *Hyperledger Fabric* [6] is an example of a framework implementing a DL in a permissioned environment. It supports Turing-complete smart contracts (as Ethereum) and follows an architecture where modules and algorithms (e.g. consensus algorithm used) can be easily

interchanged. Permissioned DLs can support an order of magnitude fewer peers participating in the consensus protocol, compared to public DLs, but offer considerably higher throughput of state transitions.

2.2 Related work

There are many cryptographic authentication and authorization systems proposed in literature. Here we succinctly review some historical widely used representatives and some modern proposed approaches employing DLs.

Traditional trust management systems: Today, digital authentication (and as an extension authorization) is performed primarily through X.509 certificates [12] issued by trusted certification authorities (CAs), therefore the infrastructure is centralized. Another centralized authorization mechanism is the Kerberos protocol [21] which employs symmetric key cryptography and requires a trusted third party. There also exist a variety of decentralized authentication and authorization systems, like the well known PGP web of trust [30], Blaze et al.'s *Decentralized Trust Management* systems [4, 5], SD3 [13] a trust management system with proofs of correct answers (decision corresponds to the specified security policy), and SPKI/SDSI [8, 23] which uses access control lists and does not require CAs. Systems for experience based trust calculation generally follow either the eigenvalue approach of *Eigentrust* [15], or the trust chain approach of [14] or [22]. A system that uses reputation to make access control decisions is presented in [26].

DLs in authentication/authorization: There is a good amount of recent work on utilizing DLs to construct more secure systems or bring transparency to existing ones. First, the conceptual advantages of using DLs for authentication are explored in [1]. The idea of a decentralized public key infrastructure (PKI) based on blockchain technology is presented in [10], and a more mature design, in the form of ClaimChains [17] can be deployed in various forms and offers enhanced privacy guarantees. Furthermore, *Blockstack* [2] offers a global naming and storage system based on the Bitcoin blockchain, while Matsumoto et al. created *IKP* [19], a system that uses Ethereum smart contracts to penalize misbehaving CAs. Finally, *WAVE* [3] is a smart contract-based authorization system for the IoT that stores access delegations in the Ethereum blockchain.

3 A STRAW-MAN SOLUTION

In this section we present an educational example of a secure trust management system which will serve as the basis for our exploration.

Environment. As seen in Fig. 1, our environment consists of the following entities, each of which is characterized by a public-private key-pair:

- *Miners:* These nodes run the consensus algorithm and execute all smart contracts, in order to generate a global and correct view of the state of the system. They are assumed to be constantly connected to the internet and have adequate processing power to run the smart contract code and, in the case of Proof-of-Work consensus algorithms, contribute to the network's processing power. They are incentivized to invest resources by receiving cryptocurrency fees that are attached to the smart contracts.
- *Internet Access Points (IAP):* Devices connected to internet which act as routers for other embedded devices. They can also act as full *validating* nodes of the distributed ledger.
- *Embedded (smart) devices:* The subject of our authorization problem. These are *things* with limited computational resources, and additionally volatile internet connectivity behaviour. They depend on the IAPs to communicate. Additionally, they are assumed tamper-resistant, and they have been initialized with knowledge of the public key(s) of their owners, i.e. the public keys that are trusted by the device to issue delegation of access credential to other public keys.
- *Users:* Users can access the internet on demand (e.g. through their 4G smartphone devices). They are the ones deciding on, and issuing delegation of access credentials.

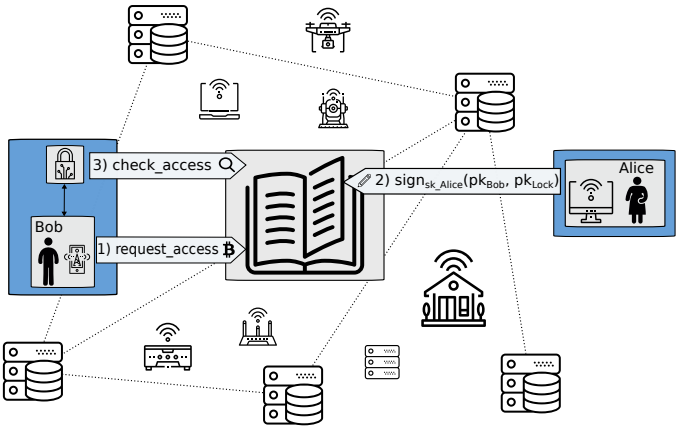

Figure 1: Our straw-man solution[2]

An example. In Figure 1, we see the general idea of an access delegation by Alice to Bob, in order for the latter to use a smart lock, to access Alice's garage for rent. Initially, Bob requests access to the smart lock from Alice, who is the owner of the lock. Alice then decides on whether or not to issue an access delegation credential to Bob's public key pk_{Bob}. The decision can be facilitated, either by personal interaction, i.e. Alice knows Bob and they have exchanged public keys through a different channel, or by querying the DL for the reputation score of pk_{Bob}, derived through past experiences of other users with Bob. After Alice decided to allow access to Bob, she sends an access control delegation, as a transaction, signed by her public key to the DL. This transaction can also demand a monetary fee in cryptocurrency to be paid by Bob, to an address associated with the delegation. Afterwards, Bob presents the address of the delegation in the DL to the system of the lock. The latter inspects the DL and assures that the delegation exists, has not been revoked, and Bob has paid the required fee to the designated address, thus allowing access to Bob, who presents his public key to the lock via a short-range communication channel (e.g. Bluetooth).

Discussion of the design. The design presented in this section is based on the property that all access delegations are stored in

[2]Design contributors acknowledgement [18]

a global distributed ledger. This design offers some interesting security properties, addressing the shortcomings of traditional trust management systems. In more detail:

(1) Complex policies, like exclusive access control can be implemented by delegation logic encoded in smart contracts. For example, to assure Bob that he is the only one with access delegation rights to the lock, delegation rules that guarantee that two overlapping delegations of the form $sign_{sk\_Alice}(pk_{Bob}, pk_{Lock}), sign_{sk\_Alice}(pk_{Charlie}, pk_{Lock})$ will be accepted as valid and added to the DL. This is the equivalent of protection against the *double spending* attack[3] against decentralized cryptocurrencies.
(2) A reputation score for each public key is maintained, and therefore a source of experience-based trust when there is no personal interaction between users.
(3) Due to the global state made available by the DL, "negative" trust evidence, like negative reputation ratings and blacklisting rules of the form: allow access to everyone except Bob, can be securely implemented, without the need for interaction between the embedded device (which may be offline) and its owner.
(4) Time-limiting rules of access can be implemented without the need of a system-external trusted global clock, by leveraging the updating (block creation) process of the DL. For example, a delegation can be valid until a chain of x state updates (new blocks) have become visible to the network. In Bitcoin, for example, the protocol enforces (through PoW difficulty adjustment) an average time between mined blocks of 10 minutes.
(5) Financial incentives and penalties can be applied natively. For example, Alice can request that Bob makes a safety deposit, unlockable by a set of arbiters. The arbiters can be chosen by their reputation score and their physical proximity to the embedded device in question. In the case that Bob damages the device, he will be liable to lose his deposit.

Why this cannot work. However attractive a design like the one described above can be, there are serious technical considerations that render it unrealizable. First and foremost, the design is unscalable. All delegations have to be included in the DL and processed by the miners. With a large number of delegations, this will overload the network. Here it is worth noting two things: (a) not all delegations may require the strict guarantees provided by a DL, and (b) the time delay between the issuance of a delegation, and its definite inclusion in the DL may not be tolerable in some scenarios. Apart from this, there are security considerations as well. In the scheme described above, devices are assumed to have access to the state of the DL. To achieve this, all devices must have adequate computational resources and disk storage, in order to store and validate the complete state of the DL. This assumption is unrealistic for embedded devices, and solutions following the direction of light clients [28] are necessary.

[3]https://en.bitcoinwiki.org/wiki/Double-spending

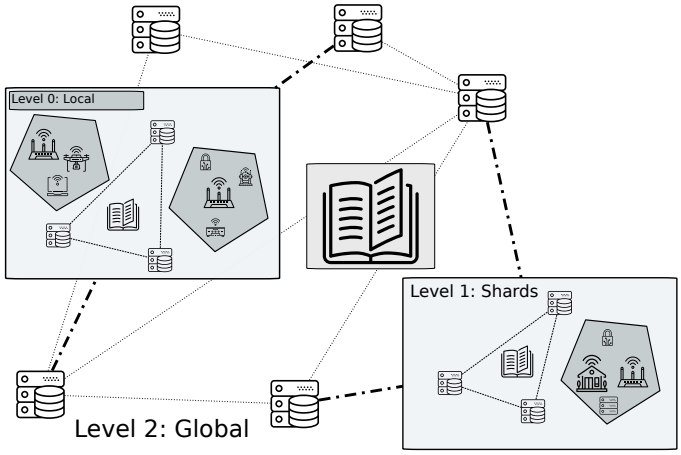

Figure 2: A 3-layer trust management architecture

4 TOWARDS SCALABLE SECURE AUTHORIZATION

In this section, we propose a layered architecture for trust management, as seen in Figure 2, and proceed to analyze choices and considerations in each layer, with an emphasis on important problems that still need to be solved, and directions towards their solution.

4.1 Layer2: Global Layer

The *global layer* represents the backbone of the system. It consists of miners maintaining a global, public DL, and it can be instantiated upon existing DLs, like Ethereum. Miners are incentivized to invest computational power with fees paid for each operation on the DL. This global DL gives us the power to enforce strong consistency guarantees for access delegations between devices and users who belong to different Layer1 shards. However, each operation on the Layer2 DL incurs a non-negligible cost w.r.t. fees that need to be paid to the miners. Apart from that, employing state-of-the art consensus algorithms, like Proof-of-Stake (which is in the roadmap of the Ethereum foundation), will remove the need for wasteful, unfair[4] and environmentally unfriendly PoW mining.

One of the basic functionality of the global layer is to keep track of the public keys of entities (users and devices) participating in the system, and the unique shard (see below) that they belong to, thus providing a prerequisite for the security properties achieved by our straw-man design.

4.2 Layer1: Shards Layer

Each *shard* or *group* belonging to Layer1 corresponds to a "smaller" DL that provides the logic for governing a specific subset of embedded devices and users (generally public keys). Conceptually, we envision this layer to correspond to a geographical "neighbourhood", like a city, or a state. Each device belongs to a unique shard at any given time, and this strong global guarantee is enforced by

[4]Proof-of-Work mining is not fair, considering that big initial investments are required to buy and set up mining "farms", and therefore the process is not open for regular users. In other words, the "voting power" of an individual scales exponentially rather than linearly with her investment.

maintaining a record of all public keys of devices that belong in each shard, on the Layer2 global ledger. On this layer, access delegations and reputation scores for entities (devices and users) subscribed to the shard are maintained. This means that the operation volume of the DL of Layer2 is, on the one hand considerable, and on the other hand affects only a part of the network. Therefore, to achieve consensus in this layer, *permissioned* DLs, employing BFT consensus algorithms are a good match. The election of the consensus participants, and their incentives for validating the contents of the DL and enforcing strong security guarantees, is a matter often overlooked in the permissioned blockchain literature. Members of the consensus group (their number is envisioned to be in the tens) can be initially randomly selected by facilitating the global DL for a bootstrapping phase, and then, they can be elected in well-defined terms based on in-system factors, such as their reputation in the system. To provide incentives to the users to participate in a consensus group, members of this group will be allocated a "compensation", in the form of a small extra fee on Layer2 transactions. The compensation will be provided at the end of their term, and will depend on the number of devices currently subscribed in their shard.

The access control process between parties and devices belonging to the same shard follows the logic of the straw-man design, presented above. However, operations concerning entities belonging to different shards may require the use of the global DL, in order to enforce policies that require access to the state of other shards.

4.3 Layer0: Local Layer

Local layer instances correspond to sets of devices in proximity. To define proximity, we assume that devices belonging to the same instance access the internet through the same access point; that is, a Layer0 instance comprises a single internet access point, and a set of embedded devices connected to it. This definition is not restrictive, as devices can freely move between access points, and is used for presentation purposes. Devices of this layer are not assumed to be constantly reachable through the internet, and generally possess limited computational resources. As a result, they depend on access points for contact with the upper layers. As a first step, we can assume an honest access point that is a full validating node of both the Layer1 DL it belongs to, and offers light client services to requesting devices. However, the trustworthiness of the access point is crucial for the security of the scheme, and monitoring mechanisms should be employed by the embedded devices, in order to assure that the access point is relaying information honestly. Each embedded device is concerned with entries of the DL concerning its public key, and according to the specified policy, decides whether or not to accept an access request.

Another aspect that has to be taken into account, and can be included in the specification of access control policies, is *context*. Specifically, embedded devices that are equipped with sensors, can sense various properties of their surroundings, like temperature, location, or whether or not they are in proximity with other devices, and restrict (or facilitate) access based on this information. For example, a policy for allowing access to a rental car with geographical restrictions (e.g. only drive in the city) can be enforced.

4.4 Hybrid authorization - delegation types

As mentioned before, not all trust delegations may require the strict security guarantees provided by including the delegation in the DL. In our architecture, we allow both *on-ledger*, and *off-ledger* delegations. For example, Alice can issue and send to Bob access credentials that would allow Bob to access a device D. Then, upon being presented the credentials by Bob, the device will first consider the state of the DL (e.g. if another user has exclusive access to the device during the time window under question), and then enforce its policy to allow access to Bob. The possibility of off-ledger delegations, when there is no need for global view guarantees, decreases the load of the ledger and allows flexible trust management. On the other hand, delegation types and policies that require global view of the state, in order to be enforced, e.g. exclusive access, or access revocation, can only be granted on the ledger. In the end, it depends on the policy specification of each device, whether or not access can be granted through off-ledger credentials.

4.5 Experience-based trust

As recounted in previous sections, the need for experience-derived information (commonly referred to as reputation), both when entities decide whether or not to request access to a resource (the usual case), but also when the entity with the capability to delegate access to a resource, or the resource (device) itself, decides on whether or not to do proceed. In this aspect, it may be important to consider own past experiences, and also past experiences of other entities, concerning the behavior of the concerned party. Traditional decentralized reputation systems, as well as flow-based decentralized reputation systems, like *Eigentrust* are either susceptible to a single point of failure, or cannot handle negative evidence and bad network connectivity. Therefore, we propose to incorporate ratings by the interacting parties, as a part of the access delegation process that is achieved through the DL. This trust evidence (in the form of ratings), can then be leveraged by mathematically tools, known in literature as *computational trust* models, that offer more informative representations of trust, compared to the simple average ratings offered by traditional reputation systems. For example, *Subjective Logic* [14] and *CertainTrust* [22] are two probabilistic models that can process evidence and output expectations regarding the trustworthiness of entities, taking into account uncertainty and conflict in the provided evidence.

5 DISCUSSION AND RESEARCH DIRECTIONS

In this paper, we revisited the problem of distributed trust management, this time in light of the IoT, Due to its scale, autonomous nature, and the connectivity and resource restrictions of embedded devices, the IoT poses new challenges for authorization. We discussed limitations of existing TM systems, and how some of these can be overcome by utilizing DLs, which can to some extent simulate a TTP, through consensus algorithms.

As a first step in our line of thought, we presented a simple architecture, where all access delegations, as well as the reputation scores of all participants are maintained in a DL, such as Ethereum. We documented the advantages of this design, for instance that it can handle exclusive access policies, time limiting rules, and "negative" trust evidence in a transparent manner. However, our proposed

design also suffers from some issues that restrict its applicability to real world scenarios. First and foremost, the design cannot scale to global dimensions, due to the load put on the DL, which has to keep the global state of all access delegations concerning all devices that take part in the system. Furthermore, the inherent overhead and time delay of encoding access delegation on a DL, is not always necessary. There are a lot of situations, where the access policies of embedded devices do not require global knowledge of the state of the network.

Towards a scalable solution that retains very strong security guarantees, we propose a "sharding of authorization domains" approach, where devices and users belong to at most one shard at any given time (a property enforced by a global DL), and the access delegations concerning devices belonging to a given shard, are maintained by a dedicated, premissioned DL. We partitioned the design space into 3 layers, namely a global, a shards, and a local layer, and described how devices are organized in these layers. Moreover, we supported the necessity to follow a hybrid approach, with the possibility to issue and exchange delegations both on and off the DL. We also highlighted the importance of experience-based trust, as a basis for decision-making, and how experience-based trust can be handled in a secure and actionable way.

Although we believe that our design is a good step towards scalable and secure TM, there are multiple challenges that need to be addressed, some of which are documented below:

– The management of delegations concerning users and devices registered in different shards, should be handled in a secure and scalable manner, taking into account the technical progress of the community in the related field of *blockchain sharding*.

– The election and formation of the consensus core of the permissioned DLs for each shard should be achieved in a way that distributes trust among participants as much as possible, and is resilient to attacks.

– A way of automatic translation of access policies is required, that translates access policies given in some specification language, to a representation that specifies what part of the trust evidence can be provided upon request, and which are included in the DL.

– The trustworthiness of the internet access point should be considered, and ways for the devices to collectively monitor and validate that it provides them with the correct and up-to-date state of the DLs should be developed.

– The incentives of the participants and the business model of the design should be explored, e.g. through game-theoretic techniques.

– The tamper-resistance of embedded devices is an assumption of our design, however the issue is of great importance on its own.

– The practical question of how to present credentials and interact with devices, from a usability perspective is a very important one on its own. One possible solution, would be to use handheld devices of users to manage their keys. This opens up a number of issues regarding key management, revocation etc. that are known to the security community, however they have not been addressed yet to the necessary extent.

To sum up, in this paper we presented an approach towards scalable and secure trust management on a global scale, and recounted inherent challenges towards its implementation. We believe that our sketched proposal is a good step for the community to critically examine and evolve.

ACKNOWLEDGEMENTS

This work has been co-funded by the DFG as part of project S1 within the CRC 1119 CROSSING.

REFERENCES

[1] Nikolaos Alexopoulos, Jörg Daubert, Max Mühlhäuser, and Sheikh Mahbub Habib. 2017. Beyond the Hype: On Using Blockchains in Trust Management for Authentication. In *Trustcom/BigDataSE/ICESS, 2017 IEEE*. IEEE, 546–553.

[2] Muneeb Ali, Jude C Nelson, Ryan Shea, and Michael J Freedman. 2016. Blockstack: A Global Naming and Storage System Secured by Blockchains.. In *USENIX Annual Technical Conference*. 181–194.

[3] Michael P Andersen, John Kolb, Kaifei Chen, Gabriel Fierro, David E Culler, and Raluca Ada Popa. 2017. *WAVE: A decentralised authorization system for IoT via blockchain smart contracts*. Technical Report. https://www2.eecs.berkeley.edu/Pubs/TechRpts/2017/EECS-2017-234.pdf

[4] Matt Blaze, Joan Feigenbaum, and Jack Lacy. 1996. Decentralized trust management. In *Security and Privacy (SP), 1996 IEEE Symposium on*. IEEE, 164–173.

[5] Matt Blaze and Angelos D Keromytis. 1999. The KeyNote trust-management system version 2. (1999).

[6] Christian Cachin. 2016. Architecture of the Hyperledger blockchain fabric. In *Workshop on Distributed Cryptocurrencies and Consensus Ledgers*.

[7] Miguel Castro, Barbara Liskov, et al. 1999. Practical Byzantine fault tolerance. In *OSDI*, Vol. 99. 173–186.

[8] Carl M Ellison. 2011. SPKI. In *Encyclopedia of Cryptography and Security*. Springer, 1243–1245.

[9] Aeleen Frisch. 2002. *Essential system administration: Tools and techniques for linux and unix administration.* " O'Reilly Media, Inc.".

[10] Conner Fromknecht, Dragos Velicanu, and Sophia Yakoubov. 2014. A Decentralized Public Key Infrastructure with Identity Retention. (2014).

[11] Yossi Gilad, Rotem Hemo, Silvio Micali, Georgios Vlachos, and Nickolai Zeldovich. 2017. Algorand: Scaling byzantine agreements for cryptocurrencies. In *Proceedings of the 26th Symposium on Operating Systems Principles*. ACM, 51–68.

[12] Russell Housley, Warwick Ford, William Polk, and David Solo. 1998. *Internet X. 509 public key infrastructure certificate and CRL profile*. Technical Report.

[13] Trevor Jim. 2001. SD3: A trust management system with certified evaluation. In *Security and Privacy (SP), 2001 IEEE Symposium on*. IEEE, 106–115.

[14] Audun Jøsang. 2001. A logic for uncertain probabilities. *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems* 9, 03 (2001), 279–311.

[15] Sepandar D Kamvar, Mario T Schlosser, and Hector Garcia-Molina. 2003. The eigentrust algorithm for reputation management in p2p networks. In *Proceedings of the 12th international conference on World Wide Web*. ACM, 640–651.

[16] Aggelos Kiayias, Alexander Russell, Bernardo David, and Roman Oliynykov. 2017. Ouroboros: A provably secure proof-of-stake blockchain protocol. In *Annual International Cryptology Conference*. Springer, 357–388.

[17] Bogdan Kulynych, Marios Isaakidis, Carmela Troncoso, and George Danezis. 2017. ClaimChain: Decentralized Public Key Infrastructure. *arXiv preprint arXiv:1707.06279* (2017).

[18] Linector, Smashicons, Najdenovski Zlatko, Pixel perfect, and Dave Freepikand Gandy. [n. d.]. Flaticon. ([n. d.]). https://www.flaticon.com/.

[19] Stephanos Matsumoto and Raphael M Reischuk. 2017. IKP: turning a PKI around with decentralized automated incentives. In *Security and Privacy (SP), 2017 IEEE Symposium on*. IEEE, 410–426.

[20] Satoshi Nakamoto. 2008. Bitcoin: A peer-to-peer electronic cash system. (2008).

[21] B Clifford Neuman and Theodore Ts'o. 1994. Kerberos: An authentication service for computer networks. *IEEE Communications magazine* 32, 9 (1994), 33–38.

[22] Sebastian Ries. 2007. Certain trust: a trust model for users and agents. In *Proceedings of the 2007 ACM symposium on Applied computing*. ACM, 1599–1604.

[23] Ronald L Rivest and Butler Lampson. 1996. SDSI-a simple distributed security infrastructure.

[24] Ravi S Sandhu, Edward J Coyne, Hal L Feinstein, and Charles E Youman. 1996. Role-based access control models. *Computer* 29, 2 (1996), 38–47.

[25] Ravi S Sandhu and Pierangela Samarati. 1994. Access control: principle and practice. *IEEE communications magazine* 32, 9 (1994), 40–48.

[26] Vitaly Shmatikov and Carolyn Talcott. 2005. Reputation-based trust management. *Journal of Computer Security* 13, 1 (2005), 167–190.

[27] Alin Tomescu and Srinivas Devadas. 2017. Catena: Efficient non-equivocation via Bitcoin. In *Security and Privacy (SP), 2017 IEEE Symposium on*. IEEE, 393–409.

[28] Ethereum wiki. 2017. Light client protocol. (2017). https://github.com/ethereum/wiki/wiki/Light-client-protocol

[29] Gavin Wood. 2014. Ethereum: A secure decentralised generalised transaction ledger. *Ethereum Project Yellow Paper* 151 (2014).

[30] Philip R Zimmermann. 1995. *The official PGP user's guide*. MIT press.