A blockchain-based Trust System for the Internet of Things

Roberto Di Pietro
HBKU-CSE
rdipietro@hbku.edu.qa

Xavier Salleras
UPF
xavier.salleras@upf.edu

Matteo Signorini
Nokia Bell Labs
matteo.signorini@nokia.com

Erez Waisbard
Nokia Bell Labs
erez.waisbard@nokia.com

ABSTRACT

One of the biggest challenges for the Internet of Things (IoT) is to bridge the currently fragmented trust domains. The traditional PKI model relies on a common root of trust and does not fit well with the heterogeneous IoT ecosystem where constrained devices belong to independent administrative domains.

In this work we describe a distributed trust model for the IoT that leverages the existing trust domains and bridges them to create end-to-end trust between IoT devices without relying on any common root of trust. Furthermore we define a new cryptographic primitive, denoted as *obligation chain* designed as a credit-based Blockchain with a built-in reputation mechanism. Its innovative design enables a wide range of use cases and business models that are simply not possible with current Blockchain-based solutions while not experiencing traditional blockchain delays. We provide a security analysis for both the obligation chain and the overall architecture and provide experimental tests that show its viability and quality.

CCS CONCEPTS

• **Security and privacy → Access control**; **Distributed systems security**;

KEYWORDS

IoT, Blockchain, Distributed Ledger, Access Control, Security

1 INTRODUCTION

The true potential of the Internet of Things (IoT) will be unleashed when billions of devices are be connected to the Internet, and able to interact with each other. However, while it is true that more and more devices are becoming connected [4], the grand vision of IoT is still far from being achieved since these devices do not communicate with each other mainly due to a lack of trust between devices, which is essential for establishing secure communication. Indeed, the trust model that works well for the Internet does not fit the scale and diversity of the IoT, where there is no common root of trust. Instead, we see different domains in which manufacturers create a root of trust that allows devices within each single domain to communicate securely. We refer to these domains as *Islands of Trust*, where the trust is provided and regulated by an entity independent from those administering other domains.

2018. Personal copy of the authors - paper accepted version. The definitive Version of Record will be published in the *Proceedings of The 23rd ACM Symposium on Access Control Models & Technologies (SACMAT '18)*, https://doi.org/10.1145/3205977.3205993.

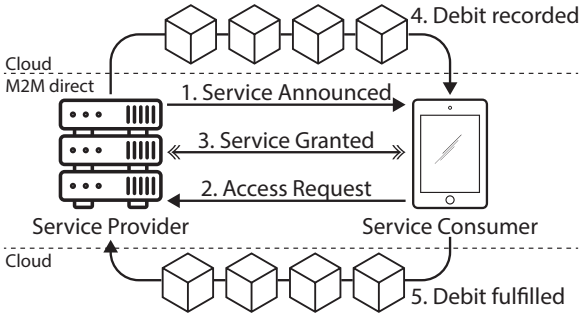

Figure 1: Our solution in brief.

Industry consortia such as the Open Connectivity Foundation (OCF) [8] attempt to solve this problem by agreeing on a common root of trust but do not cover the entire IoT landscape. As a result, the current IoT consists individual manufacturers and platforms that can communicate securely only if they agree on a common root of trust (e.g. through a consortium) or if they establish direct mutual trust through bilateral agreements.

In this work we take a new approach for bridging trust between the above domains (i.e. the islands) by leveraging blockchain technologies [5] to create a distributed trust mechanism. We start by introducing a new tool, named *Obligation Chain*, which is a new platform for a distributed credit-like system (in contrast to the cash-like Bitcoin [11]). Furthermore, our credit system has a built-in reputation mechanism [13] that allows peers to decide whether or not to accept obligations based on the credit history of a consumer. Figure 1 shows an overview of our main scheme in which a service provider interacts with a service consumer providing real time services while postponing the obligation fulfillment.

The benefits of our construction can best be seen through the following use case: a service provider offers a service, together with its terms of use. For example, let us consider a small coffee-shop that wishes to offer WiFi services for customers (for an additional fee). Currently, major operators and major coffee chains have bilateral agreements that allow this, but there is no solution that fits any small/family run coffee-shops. We would like to have a solution that allows anyone to consume these services simply by providing a public obligation for fulfilling the terms of use as specified by the service provider. Following good practices, the service provider is not expected to automatically accept an obligation from anyone, but it is expected to first assess the risk of accepting that obligation. Naturally, a large part of potential consumers might not

have enough reputation to access the service. This is indeed where many reputation systems fail in practice [9] and this is why our system goes a step further to leverage existing trust. In the above example, we have leveraged the trust these users already have with their mobile operator and the fact that the mobile operator already has a well established reputation. Furthermore there is already a full fledged public key infrastructures (PKIs) in place mediating between the mobile operator and its customers. By bridging trust between the coffee shop and the mobile operator, we are able to provide a complete path of trust between any customer of the mobile operator and the service provider.

The coffee shop simply needs to publish the terms of use for accessing the WiFi service. Any large mobile operator that wishes to grant free access of the WiFi services to its customers can sign an obligation to fulfill the terms of use. The users would get the signed obligation based on their existing trust relationship with their operators. The users would then present the signed obligations to the coffee shop that would need to decide whether or not to grant access based on the public credibility of the signing operator. As the entire history of obligation fulfillment is available on the immutable blockchain, the service provider can make its decision of whether or not to accept the obligation. If the obligation is accepted and later fulfilled then the service provider would report it on the blockchain. By doing so, it would increase the operators' credibility.

The solution described in this paper allows every service provider to conduct its own assessment based on the credit history of the obligation's signer, the service value and any other element it could deem relevant. Thus, one service provider may decide to accept an obligation from a certain consumer, while another one may not. Furthermore it has the flexibility to offer complex business models that may depend on the actual consumption of the service while requiring minimal capabilities from the end devices.

The rest of this paper is organized as follows. In Section 2 we summarize the building blocks leveraged in this paper. In Section 3 we introduce the core elements and main concepts defined in this solution, paving the way to Section 4, where we describe our solution. In Section 5 we provide a security analysis of our solution and compare its performances against the well-known Bitcoin system (the measurements are done over our reference implementation[1] that was developed in the context of the production of the paper[6] and is available on GitHub). Section 7 concludes the paper.

2 BACKGROUND

Trust is generally perceived as a belief that an entity is honest and will not harm other entities. This belief is subjective and based on past experiences. On the other hand, reputation is a global perception of an entity's behavior based on the trust that other entities have established [13]. The goal of a Trust and Reputation System (TRS) is then to guarantee that actions taken by entities in a system reflect their reputation values and cannot be manipulated by unauthorized entities [2, 3, 12]. TRSs can be generalized as composed by entities, observers, disseminators and reputation servers [9] and have been shown to be threatened by different attacks [9]. In this paper, we focus on how to design a blockchain based TRS which allows for bridging of trust between secure domains.

[1] https://github.com/xevisalle/IslandsOfTrust

2.1 Blockchain Technology

In the last few years, a new technology named *Blockchain* [5, 18] which first emerged with Bitcoin [11], has had great success in many areas. This technology can be roughly described as a digital ledger that sits at the core of decentralized ecosystems and keeps track of any changes by holding a new record for each transaction. In a more abstract way, the blockchain can be seen as an ordered and back-linked list of blocks carrying transactions which encode exchanged information between two or more participants. Each block consists of a collection of transactions and is linked to the previous block in the blockchain, thus creating a chronological order of blocks that all together build a chronological order of transactions.

3 SETTINGS AND DEFINITIONS

The main goal of our solution is to build an access control system that leverages consumers' reputation and that is used by service providers. To this end, our basic setup is made by a *service provider* (SP) and a *service consumer* (SC). Given a SCD seeking to access some service from a SPD we seek to establish trust between the SCB and SPB and then pass that trust down to the devices. The scenario can be enriched in complexity, but for the sake of exposition we will stick to the simple model just introduced. Each of the two entities has access to powerful *back-ends* and more constrained *end-devices* (respectively SPB/D and SCB/D). All internal communications inside the SP and the SC are secured leveraging the existing trust within the same Island of Trust (i.e. same domain) while external communications between SP and SC are secured using our solution.

Symbol	Description
SP	service provider
SPD	service provider end-device
SPB	service provider back-end server
SC	service consumer
SCD	service consumer end-device
SCB	service consumer back-end server

Table 1: Table of symbols

SC and SP has a self-generated public key, used to sign messages and as an ID, recognized by others. The reputation is associated with this ID. We stress that in our solution there is no certification authority (CA) that manages these IDs. Instead, they are self-generated within each Island of Trust and may be replaced at any time. However, as explained in the rest of this paper, both SCs and SPs are usually keen to keep the same IDs as needed to claim their reputation. SPs and SCs mainly cooperate by exchanging *terms of use* (TERMS) and *obligations*. The former are created by service providers and signed by both service providers and consumers. They are here intended as rules that SCs must agree to abide to in order to access services provided by SPs. The latter are tokens leveraged by SCs to publicly state that all the conditions listed within TERMS will be fulfilled as *per* what has been specified by SPs.

As usual in life, building a good reputation is much harder than ruining it, which mitigates the attack where an SC builds a good reputation over a long time only later sign an obligations it does not intend to fulfill. We note that reputation is built on honored obligation thus wandering on the security of the bootstrap phase. Luckily, bootstrapping in trust and reputation management systems has been extensively treated in the literature in the past (for instance, in the context of P2P systems). And the topic is also being addressed by leveraging blockchain technologies [1].

4 OUR SOLUTION

We now show how TERMS and obligations can be used by SPs and SCs to set up trusted interactions between untrusted devices. To this end, we have designed a new blockchain named *obligation chain* that is linked to to another blockchain and used to build a tamper-proof reputation system. In this section we first introduce this new chain and then describe how we have leveraged it to bridge the different islands of trust (i.e. different domains).

4.1 Obligation Chains

Being based on the blockchain technology, our obligation chain is eventually agreed upon by the whole network. It can be seen as a distributed ledger storing obligations of commitments signed by SCs to access SPs' services without immediately paying for them. For the sake of simplicity, we can imagine our obligation chain as an append only log database where obligations and TERMS are kept. However, unlike other solutions, our chain does not contain digital assets which have to be recognized and verified by other peers at run-time (as for bitcoins in the Bitcoin blockchain). Indeed, our obligation chain contains obligations that are *locally* accepted by SPs and then shared to the rest of the network via the same SP.

Obligations are generated by SCBs and initially contain only the TERMS and their signature. They are then downloaded to SCDs which will later exchange them, in the form of an obligation transaction, to get access to services provided by SPs. Obligation transactions contain, among others: the TERMS which have been previously published by SPBs and agreed to by SCBs, a unique ID of the obligation and all the signatures provided by SP/SC entities.

Obligation transactions are leveraged to keep track (on the chain) of the agreements that are established (off the chain) between SCs and SPs. As such, as both SCs and SPs built their reputation on the chain, they both need tools to prevent fraudulent interactions. On the one hand, the proof of commitment described above is the tool used by SPs against malicious SCs. On the other hand, the *proof of fulfillment* is the tool we have designed for honest SCs which have to protect themselves against malicious SPs.

The proof of fulfillment has been realized by linking our obligation chains with standard blockchains. Obligations' fulfillment is not public since obligations are accepted locally by SPs. As such, a TERM signed by SP_i and SC_j serves as a proof that SC_j owes some money to SP_i. Nothing else is needed by others SPs to decide on SC_j reputation. It would have been also possible to use smart contracts within a single blockchain. However, although using smart contract is indeed a possibility that does not change the framework set by the proposed solutions, our solution does not require smart contracts which are more costly and limited to certain platforms.

For the sake of simplicity, we will consider a toy example in which services provided by SPs need some kind of payment in order to be accessed. To this end, throughout the rest of the paper, we have put forward the Bitcoin solution as a concrete example to cite. However, as we describe in the remaining of this section, our solution is agnostic with respect to the payment solution and the adopted technology and it can be applied to any other blockchain. Furthermore, it is also important to highlight that the validity of an obligation and the acceptance of the transaction containing that obligation are different concepts. Obligations are accepted if signed appropriately by the participating parties. Namely, by the SCB, SCD and SPD. If the transactions is created and signed by SP_i, since the obligation validity is locally taken, then the transaction per-se is considered valid.

As detailed in Section 4.2, the update of SCs' reputations might be synchronous and/or asynchronous. On the one hand, in the asynchronous approach, SPs cannot always remain updated on both the obligation and the Bitcoin blockchains. As such, they connect to them and download new blocks only when they need to update service consumers' reputation scores. As an example, the first time that a given service consumer C approaches a service provider requires the latter to read the obligation and bitcoin chains from their origin in order to bootstrap C's reputation. Contrariwise, for all those service consumers for which a service provider does not have updated info on their reputations, only the new blocks need to be accessed. On the other hand, in the synchronous (or Cached) approach, SPs always receive the latest obligation and Bitcoin blocks in the chains. As such, there is no need to build the local reputation scores for their service consumers from scratch. Hence, given a service consumer C only new blocks in which C's obligations and payments are stored will need to be downloaded.

4.2 Islands of Trust

Now that we have described the obligation chain and how it is leveraged to store SCs' obligations, we can introduce the concept of *bridging the trust* between different islands of trust, i.e. different secure domains. In our solution, we assume that each island of trust has a full local PKI and CA. Such a PKI is not recognized or trusted by other islands and it is only used for the internal device management. However, PKI information (such as the public keys) are used as publicly available IDs. Reputations are built on top of such IDs. Furthermore, we assume that each device has a certificate that was issued by its local CA and that is used to secure the communications between end-devices and back-ends within the same island. Each device belonging to either the SPB or the SCB can be uniquely identified by them thanks to unique IDs.

The protocol that we have designed for the trust bridging is a three-way handshaking protocol composed by *setup*, *spend*, and *fulfilling* phase. During the *setup*, SCBs generate (and sign) obligations based on the TERMS published by SPBs. These obligations are passed to SCDs to be used when interacting with SPDs. In the *spend* phase, end-devices coming from both SPs and SCs interact with each other and exchange their own version of TERMS. If they match, the obligations created in the *setup* phase are passed to the SPB. If the SPB decides to accept the obligations based on the local SC's reputation score, the SCD receives access to the service from

SPD and its obligations are broadcast to the rest of the network to be included in the obligation blockchain. In the final step, SCs make a connection between the Bitcoin and the obligation chain by paying for them. This final step causes all the local reputation scores within all the SPs to be updated which also increases the likelihood of SC's new obligations to be accepted in the future. It has to be noted that even though a PKI and CA is used to access the blockchain, such access can be multi-tenant (as provided by IBM Hyperledger). As such, no root of trust is needed to provide SPs with a tool to compute local SC reputation scores.

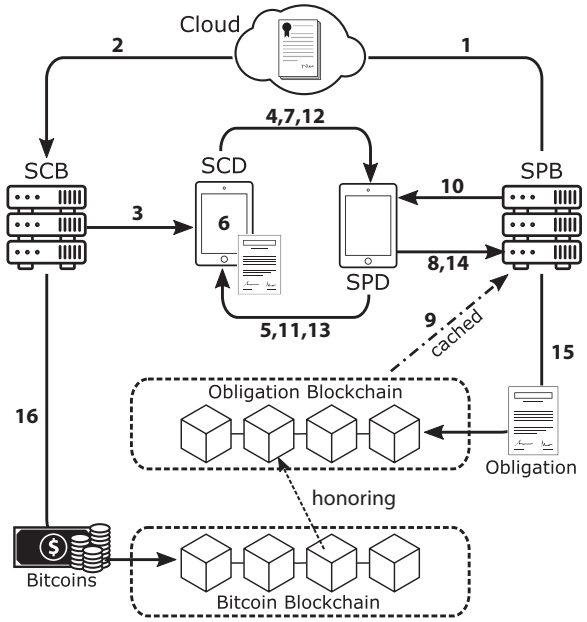

Figure 2: Protocol Overview.

Figure 2 depicts our three-way handshaking protocol and highlights each individual operation executed by both SC and SP entities (both end-devices and back-ends). In our proof of concept we implemented SPB and SCB to be miners, both for Multichain and Bitcoin. However, as for all the current blockchain-based systems, this is a matter of choice. SPB and SCB might be either miners or clients. The latter assumes other peers are maintaining the chain. The complete flow is as follows (see also Figure 3):

(1) The SPB publishes the TERMS[2] with all the required details (e.g. the price);
(2) The SCB downloads SP TERMS and creates obligations. The obligations contain all the details written within the TERMS and are signed by the SCB;
(3) The above obligations are downloaded within the SCDs that intend to use them;
(4) The SCD sends a service request to the SPD ;
(5) The SPD sends the TERMS to the SCD;
(6) The SCD compares the TERMS to the ones in its obligation to ensure they match;
(7) If the TERMS match the SCD passes the obligation to the SPD, otherwise it ABORT;
(8) The SPD sends the obligation to its SPB;
(9) The SPB looks at the obligation chain and payment chain to assess the credibility of the obligation issuer. Depending on the approach being used (which can be either *synchronous* or *asynchronous* as described above) a complete bootstrap or an update of the consumer's reputation are executed accordingly. The result is an updated knowledge of SC's trustworthiness. This result is local to the SP that is computing it and is based on an arbitrary trust score evaluation which is beyond the scope of this work. The SPB also verifies the signature w.r.t. the public key of the issuer;
(10) If the signature is valid, the TERMS match, and the SPB decides to trust the SC based on his obligations history and thus his reputation, then the SPB sends an OK to the SPD. Otherwise, it sends ABORT. SPB also generates and sends back to the SPD a new payment address. As we are considering the toy example of a service provisioning that requires Bitcoin payments, this new address will be a new Bitcoin address. This address needs to be sent back to the SCD and SCB as it will be later used to fulfill the obligation;
(11) The SPD either conveys the reply to the SCD, if it has received an OK from its SPB, or it ABORT the protocol;
(12) The SCD signs the obligation;
(13) The SCD passes the signed obligation to the SPD (this is to allow the SCB to keep track of used obligations);
(14) The SPD verifies the SCD signature. If the signature is valid, then it gives access to the service otherwise ABORT;
(15) The SPD signs the obligation just received from the SCD;
(16) The SPD sends its signature back to the SCD (to serve as a receipt of the transaction);
(17) The SPD conveys the signed obligation to the SPB;
(18) The SPB broadcasts the obligation to the other peers in the obligation chain network in order for the obligation to be added in the obligation chain;
(19) The SCB periodically monitors the obligation chain looking for its obligations which are then fulfilled by issuing payment transactions to the exact Bitcoin addresses that have been created by the SPB at step 9. This allows the SP to monitor those addresses and to detect new bitcoin incomes. The result is for the SP to remove the pending obligation from its internal database, thus also updating the SC's reputation.

It is again important to highlight that, whilst all the aforementioned steps are executed at run-time, the step number 8 can also be executed off-line in the *asynchronous* mode (see Section 4.1 for more details). In this mode, SPs can update SCs' reputations each time they receive a new block head for the obligation chain. This makes the whole process faster as SPs are already aware of SCs' reputations even before they receive new SCs' requests. During our experimental tests we have used the asynchronous mode as it represents the worst case scenario—the SP has to rebuild the consumer's reputation every time. However, even in the worst case

[2]We do not specify exactly how these TERMS are published. We envision them to be posted on the service provider's web site, but any other solution would do. We also note that the TERM may change over time and that the SCB is the solely responsible for keeping track of changes.

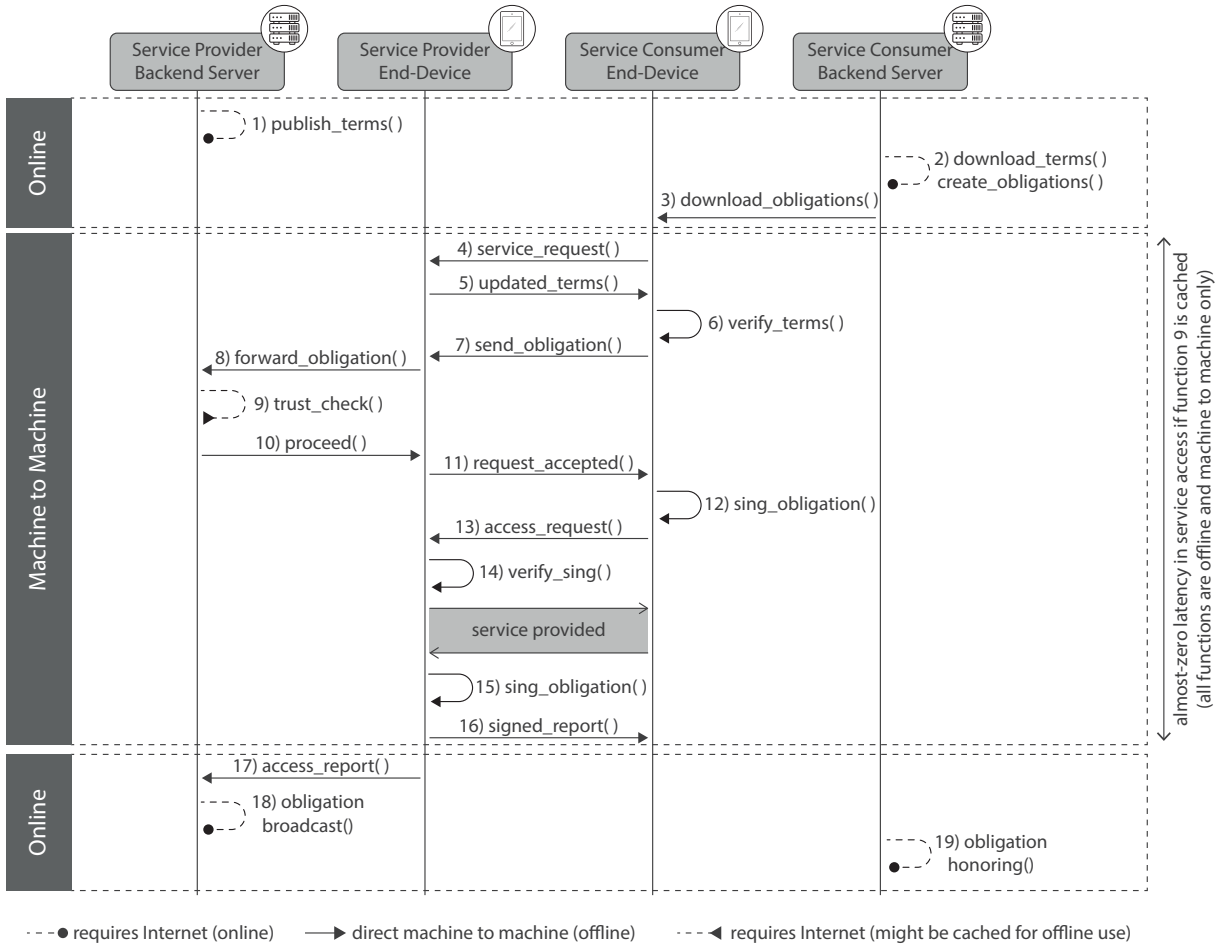

Figure 3: Protocol sequence diagram.

scenario our solutions proved to be faster than standard blockchain based payment systems such as Bitcoin.

One of the main advantages of this solution is that service providers and consumers do not have to explicitly establish a contractual agreement. SPs publish the possible business models as part of their TERMS while SCs pick the ones that best suit their needs. Then the decision on whether or not to accept the obligations is left to the SPs. Another key advantage of the proposed solution is the establishment of self-enforced and tamper-resistant reputations. Indeed, as we are assuming that SCs and SPs do not know each other in advance, they can judge each other's trustworthiness only based on the reputations stored within our obligation chain. In this work the reputation is meant as the average of obligations fulfilled on time. This information is accessible to anyone having access to both the obligation blockchain and the Bitcoin blockchain as it is only required to check how many obligations have been fulfilled by using the new Bitcoin addresses created in step 9 of our protocol. Hence, peers with a high frequency of interactions with others will create many obligations and will need to fulfill them as established with the SP. If they do so, they will have a good reputation, otherwise they will not. Furthermore, TERMS also contain the time by which the payment needs to be completed. A consumer that does not pay or that delays the payment will lose credibility and will be rejected by others in the future.

4.3 Privacy

So far we argued about the benefits of having all transactions publicly recorded as part of a distributed reputation platform. However, this also raises some privacy concerns. As such, we have used encryption to provide confidentiality of the obligations between the parties by encrypting TERMS. Naturally SPs and SCs would both have the keys to decrypt TERMS and their signatures. However, keeping the keys solely between a SP and a SC would not enable others to assess the reputation of the latter. Thus, when a SC approaches a new SP, he would need to disclose the decryption keys (which is also used nowadays when users need to present their bank records and credit history when asking for a loan).

This enables the new SP to see SC's good credit history while also allowing the latter to hide bad credit history. We solved such a problem by publishing the obligations that were not fulfilled in an unencrypted form (i.e. along with their keys) and linking them to the previous encrypted ones. The result is that some obligations

are encrypted (i.e. revealed by SCs only to the right SPs) while all the non-fulfilled obligations are public.

5 ANALYSIS

5.1 Security

Trust and reputation systems (TRS) have been suggested as an effective security mechanism for open and distributed environments. However, it has also been shown that such mechanisms can be threatened by different attacks as described by Fraga et al. [9] who identified information gathering, calculation and dissemination attacks. Based on such classification, we have analyzed attacks in our solution that might threaten i) how SPs gather information on SCs, ii) how SPs access the obligation blockchain and make decisions on SCs' reputations as well as iii) how it is possible to thwart the process in which SCs' information is shared among SPs.

As a first result it has to be noted that, all the attacks targeting observers, reputation servers, or entities in classical TRSs, do not apply here as they are part of the islands. Hence, as defined in Section 3, those elements belong to a single domain and thus assumed secure. Furthermore, even attacks based on data manipulation do not apply due to the blockchain technology. However, other classical (i.e. non TRS specific) attacks might still occur. *Privilege escalation* might be one of those in which a malicious SC creates and uses a big set of obligations to build a temporary good reputation for malicious purposes. Our solution mitigates this attack as new obligations do not affect SC's reputation until a payment is validated within the Bitcoin chain that links to it.

A complementary attack might also be unleashed in the presence of operators that refuse to endorse their clients thus producing a denial of service (DoS) attack. Although this theoretically is a possibility, it already exists in current systems (e.g. Gmail denying access to the inbox of some clients). However, in this setting it is unlikely that rational service providers will be keen on compromising their reputation by playing a DoS on their users or cheating them since their own (social) reputations could be jeopardized. Still this is an interesting point to address in future works. Ballot stuffing attacks can be easily mitigated as well via obligations' fees or watch-dog systems. This will reduce the incentive for the attacker to generate fake transactions and detect them within the chain.

A full and detailed comparison against the state of the art in TRS attacks will be given in the extended version of our paper.

5.2 Performance

In this section we present a performance analysis of our solution on the time required for SPs to compute SCs' reputations where the creation and validation of obligations have been accomplished by using the *SigningKey*, *SECP256k1* and *VerifyingKey* python functions from the *ecdsa* library. This analysis is indeed required to show if, as expected, the proposed solution provides a virtual zero-latency user experience. The above test has been conducted in the worst case scenario (the one in which SPs have to always rebuild SCs' reputation from scratch) and compared it to the Bitcoin best case scenario (i.e. the smallest block acceptance delay witnessed in the last year). To implement such a worst case scenario, we have automated 1460 handshake protocols between a SC and a SP (i.e. the Bitcoin blocks created between Oct. 2016 and Sept. 2017).

The results clearly showed that our acceptance delay is always shorter than Bitcoin even in the case in which all the 1460 new obligations are verified at the same time (we refer to the asynchronous approach described in Section 3). Indeed, checking the reputations of 1460 service consumers took as much time (6.9 minutes) as needed by Bitcoin to add a single block to the chain. Taking also into account that a new Bitcoin block usually requires one hour to be considered valid and that updating SCs' reputations in a synchronous environment (see Section 3) only required less than a second, finally shows that we have achieved a zero latency user experience.

6 RELATED WORK

Recently, in the academic world, there is a trend towards redesigning rating systems (i.e. TRSs) given the rise of the new blockchain era. As examples, Schaub et al. [15] and Soska-Christin [16] have both designed users' privacy-aware solutions based on the blockchain technology. However, the aforementioned approaches were not focused on TRS specific attacks but rather on general blockchain vulnerabilities. In this paper we are more focused on reputation attacks such as *rating frauds* in which a malicious user tries to seek inappropriate profits from the system. Such attacks occur both in content-driven and non-computational TRS systems but can be mitigated by using the blockchain technology as a source of immutable and non-repudiable rating information.

In the last few years, different proposals have been published to that end. Dennis et al. [14] proposed a solution in which the human rating factor (usually associated to *feedback*) has been removed. The result is a binary rating system in which either the service has been provided or not. Such an approach, however, weakens the rating system as it removes the *service quality* aspect. Indeed, the final goal of TRSs is to help SCs in understanding sellers. As such, if we only keep track of whether the product/service has been delivered, we will lose other factors and information which are as important as the delivery. In our solution we create commitments on the service quality which are signed by both SPs and SCs. As such, although we use a binary feedback (an obligation can be either fulfilled or not fulfilled) we maintain services' quality information within the obligation.

Other works tried to mitigate rating fraud by raising rate costs. This approach has been largely used in the past for standard TRSs. As examples, Douceur et al. prevented sybil attacks by binding accounts to unique IPs [7] while Yu et al. [17] increased the difficulty in controlling multiple accounts. Another example of raising costs or complexity has been designed by Yosang and Ismail [10]. In this solution, raters are encouraged to provide honest rates by sharing incomes with them. However, such kind of solutions are not effective if the perceived benefit from the attacks is greater than its cost. In traditional Cloud services (such as *Amazon.com* or *Expedia.com*), this problem is solved by using *verified transaction* labels but requires trust to be centralized.

The immutable and eventually agreed database held by the blockchain technology can be used to mitigate the aforementioned rating attacks. As an example, Schaub et al. [15] proposed an interesting approach against bad mouthing attacks. In their solution, they kept the user feedback while allowing only peers who actually received

tokens from seller to rate them. Compared to this approach, our solution solves the same problem but with a different perspective. Indeed, whilst Schaub et al. try to protect SCs from malicious SPs, we defend the latter from frauds. The main reason why we focus on this open challenge is that our solution is based on credits. As such, SPs accept *payment obligations* and make a decision on whether to trust the SC. Unlike what has been proposed by Schaub et al. in which SCs can take their time before making transactions with SPs, we need to take immediate (i.e. zero latency) decisions. In our scenario, SPs do not know at what time SCs will contact them and do not even know their identities. Still, they want to accept obligations while being sure to not be cozen.

TRSs can also be threatened by *ballot stuffing* sybil attacks in which the service provider colludes with SCs to gain reputations. Usually, these attacks leverage small fraudulent purchases. As such, a traditional way to mitigate them is to remove the *verified purchase* label on top of discounted transactions. As shown by Schaub et al., the blockchain technology can be used against ballot stuffing attacks as well. However, in their solution, SPs are limited in the number of tokens (feedback) that can be used. Although this approach was proved to be effective (as it creates a trade-off between ratings and profit), it limits the overall business model. In our solution we do not assume any limitation on the number of obligations that can be created by SCs and accepted by SPs.

7 CONCLUSION

In this paper we have shown a solution that enables trust establishment among devices belonging to different domains. In particular, our solution is suited for the IoT context, given its unique features of being fully decentralized, and requiring both security and negligible overhead on end devices. These features are achieved in our proposal by leveraging the idiosyncratic properties of blockchain technologies, combined with a new architecture design that avoids the pitfalls inherited by this technology, while unleashing its advantages. In particular, as the performance of our implementation shows, all the above features are achieved in a very efficient way, and significantly faster when compared to the standard Bitcoin based solution —hence enabling those use cases that otherwise could not work when facing long delays.

It is worth noting that our solution supports rich and flexible business models and use cases. Indeed, by allowing a seamless level of trust and cooperation among actors belonging to different domains, it removes entry-barriers to service providers thus introducing a disruptive degree of innovation. Finally, the obligation chain enables a wide range of credit based use cases.

We believe that the flexibility of our solution in supporting different business models, its degree of innovation, combined with its efficiency and overall deployability has a clear potential for opening further research threads in the highlighted directions.

ACKNOWLEDGMENT

We like to thank Louis Shekhtman for his help in improving the editorial quality of this paper.

REFERENCES

[1] Muneeb Ali, Jude Nelson, Ryan Shea, and Michael J. Freedman. 2016. *Bootstrapping Trust in Distributed Systems with Blockchains.* Technical Report. Blockstack.

[2] Sulin Ba and Paul A. Pavlou. 2002. Evidence of the Effect of Trust Building Technology in Electronic Markets: Price Premiums and Buyer Behavior. *MIS Quarterly* 26, 3 (sep 2002), 243.

[3] Gary E. Bolton, Elena Katok, and Axel Ockenfels. 2004. How Effective Are Electronic Reputation Mechanisms? An Experimental Investigation. *Management Science* 50, 11 (nov 2004), 1587–1602.

[4] V. Daza, R. Di Pietro, I. Klimek, and M. Signorini. 2017. CONNECT: CONtextual NamE disCovery for blockchain-based services in the IoT. In *2017 IEEE International Conference on Communications (ICC)*. IEEE, Paris, France, 1–6.

[5] Christian Decker and Roger Wattenhofer. 2013. Information propagation in the Bitcoin network. In *IEEE P2P 2013 Proceedings*. Institute of Electrical and Electronics Engineers (IEEE), Trento, Italy, 1–10.

[6] Roberto di Pietro, Xavier Salleras, Matteo Signorini, and Erez Waisbard. 2018. A blockchain-based Trust System for the Internet of Things - Extended. https://cri-lab.net/wp-content/uploads/2018/04/IslandsOfTrust-Clean.pdf

[7] John R. Douceur. 2002. The Sybil Attack. In *Revised Papers from the First International Workshop on Peer-to-Peer Systems (IPTPS '01)*. Springer-Verlag, London, UK, UK, 251–260.

[8] Open Connectivity Foundation. 2017. Online: https://openconnectivity.org/.

[9] D. Fraga, Z. Bankovic, and J.M. Moya. 2012. A Taxonomy of Trust and Reputation System Attacks. In *2012 IEEE 11th International Conference on Trust, Security and Privacy in Computing and Communications*. Institute of Electrical and Electronics Engineers (IEEE), Liverpool, UK, 41–50.

[10] Audun Josang and Roslan Ismail. 2002. The beta reputation system. In *In Proceedings of the 15th Bled Conference on Electronic Commerce*. Electronic Commerce Center, Bled, Slovenia.

[11] Sergio Martins and Yang Yang. 2011. Introduction to Bitcoins: A Pseudo-anonymous Electronic Currency System. In *Proceedings of the 2011 Conference of the Center for Advanced Studies on Collaborative Research (CASCON '11)*. IBM Corp., Riverton, NJ, USA, 349–350.

[12] Do-Hyung Park, Jumin Lee, and Ingoo Han. 2007. The Effect of On-Line Consumer Reviews on Consumer Purchasing Intention: The Moderating Role of Involvement. *International Journal of Electronic Commerce* 11, 4 (jul 2007), 125–148.

[13] Paul Resnick, Ko Kuwabara, Richard Zeckhauser, and Eric Friedman. 2000. Reputation systems. *Commun. ACM* 43, 12 (dec 2000), 45–48.

[14] Dennis Richard and Owenson Gareth. 2016. Rep on the Roll: A Peer to Peer Reputation System Based on a Rolling Blockchain. *International Journal of Digital Society (IJDS)* 7 (3 2016), 1123–1134.

[15] Alexander Schaub, Rémi Bazin, Omar Hasan, and Lionel Brunie. 2016. A Trustless Privacy-Preserving Reputation System. In *ICT Systems Security and Privacy Protection*. Springer Nature, Cham, 398–411.

[16] Kyle Soska and Nicolas Christin. 2015. Measuring the Longitudinal Evolution of the Online Anonymous Marketplace Ecosystem. In *Proceedings of the 24th USENIX Conference on Security Symposium (SEC'15)*. USENIX Association, Berkeley, CA, USA, 33–48.

[17] Haifeng Yu, Michael Kaminsky, Phillip B. Gibbons, and Abraham Flaxman. 2006. SybilGuard: Defending Against Sybil Attacks via Social Networks. In *Proceedings of the 2006 Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications (SIGCOMM '06)*. ACM, New York, NY, USA, 267–278.

[18] Guy Zyskind, Oz Nathan, and Alex' Sandy' Pentland. 2015. Decentralizing Privacy: Using Blockchain to Protect Personal Data. In *2015 IEEE Security and Privacy Workshops*. Institute of Electrical and Electronics Engineers (IEEE), San Jose, CA, USA, 180–184.