2018 IEEE 4th International Conference on Collaboration and Internet Computing

Blockchain-based PKI solutions for IoT

Ankush Singla
Department of Computer Science
Purdue University
West Lafayette, USA
asingla@purdue.edu

Elisa Bertino
Department of Computer Science
Purdue University
West Lafayette, USA
bertino@purdue.edu

***Abstract*—Traditionally, a Certification Authority (CA) is required to sign, manage, verify and revoke public key certificates. Multiple CAs together form the CA-based Public Key Infrastructure (PKI). The use of a PKI forces one to place trust in the CAs, which have proven to be a single point-of-failure on multiple occasions. Blockchain has emerged as a transformational technology that replaces centralized trusted third parties with a decentralized, publicly verifiable, peer-to-peer data store which maintains data integrity among nodes through various consensus protocols. In this paper, we deploy three blockchain-based alternatives to the CA-based PKI for supporting IoT devices, based on Emercoin Name Value Service (NVS), smart contracts by Ethereum blockchain, and Ethereum Light Sync client. We compare these approaches with CA-based PKI and show that they are much more efficient in terms of computational and storage requirements in addition to providing a more robust and scalable PKI.**

***Index Terms*—Internet of Things, Blockchain, Smart Contracts**

I. Introduction

A. Background and Motivation

The Internet of Things (IoT) is estimated to connect 30 billion devices to the Internet by 2020 [1]. Securing the enormous amount of data generated and transmitted by these devices while minimizing the computational overhead added by the enforced security measures is a huge challenge. Digital certificates are a key component of current security infrastructure as they bind a cryptographic public-key to its owning entity. Such a certificate is used to verify the identity of an entity; the public-key can then be used to establish a secure communication channel using standard cryptographic key-agreement protocols. The most common standard for digital certificates is the X.509 standard [2]. An X.509 certificate contains information like Signature Algorithm ID, Validity period, Subject name, Public Key Algorithm, Subject Public Key, Certificate Signature Algorithm, and Certificate Signature. These digital certificates form the backbone of the Secure Sockets Layer/Transport Layer Security (SSL/TLS) protocols.

Traditionally, a trusted third party (TTP) known as certification authority (CA) is required to sign, manage, verify and revoke digital certificates. Multiple CAs together manage the public key infrastructure (PKI) known as CA-based PKI. CAs are generally private entities that provide their services at a cost. They perform three important tasks for the functioning of the CA-based PKI: (i) Verifying the identities of the entities requesting their certificate be signed and creating a cryptographic signature for the certificate using their own private key. (ii) Providing users an efficient way to verify that the certificate of the party they are communicating with is valid. (iii) Maintaining a list of certificates that have been revoked for different reasons, such as private key leakage, server compromise or other administrative reasons.

All such tasks require end users to fully trust multiple CAs, and assume that they will work in a fair and benign manner and also do the due diligence when signing the certificates. However, several issues can arise due to this requirement of a trust anchor for creating and maintaining the records and the infrastructure surrounding it. The root CA or the intermediate CA private keys might be compromised due to security breaches and can be used to sign bogus certificates for a malicious party impersonating another entity. Any security breach of the trusted parties can affect large numbers of users. This problem is magnified in CA-based PKI because there are multiple TTPs which increases the surface area of attack for a security infrastructure.

Blockchain has emerged as a transformational technology for applications where a centralized TTP was traditionally required for creating and maintaining records. It replaces TTPs with a publicly verifiable, distributed, peer-to-peer, append-only data store which maintains its integrity using various consensus protocols. The use of blockchain for public-key management has been investigated in [3] [4] [5], but it has not been explored in detail in the context of IoT systems. Running a full blockchain node may require high memory and computational resources which are often limited in case of IoT devices.

B. Our Work

The blockchain approaches for PKI address several shortcomings of the CA-based PKI, while improving the performance of the verification process. However specific performance advantages depend on the blockchain technology adopted. In this paper, we thus compare three blockchain-based alternatives to the traditional CA-based PKI. To benchmark the Blockchain protocols, we modify the OpenSSL [6] cryptographic library, which is a widely used open-source implementation of the SSL and TLS protocols.

978-1-5386-9502-9/18/$31.00 ©2018 IEEE
DOI 10.1109/CIC.2018.00-45

The first approach we consider uses the Emercoin [7] public blockchain which provides a Name Value Storage (NVS) capability. We register the ID and cryptographic hash of the certificates as a (Name:Value) pair in the blockchain which can be referenced by any device for certificate verification purposes. To query the blockchain, the IoT devices connect to a remote trusted blockchain node that actually stores and maintains the blockchain. This node is a part of the blockchain network and will need to be hosted by the network managers of the specific IoT ecosystem or organization. For example, in a university setting, the network administrator hosts on a local server the blockchain node and all the IoT devices that are a part of that ecosystem, such as for example the smart sensors, parking meters, and smart locks, can query this blockchain node to extract information.

The second approach we consider uses Ethereum [8] smart contracts and enhances storage flexibility when compared to the first approach while maintaining all the benefits. Ethereum smart contracts enable storage of more complex data structures including mappings and relationships. Several rules can also be enforced on modifications of these data structures.

The first two approaches require the presence of a remote trusted blockchain node that stores the actual copy of the blockchain. The third approach we consider is based on "Ethereum Light Sync mode" and removes the need for such remote blockchain node. The IoT device just needs to store a lightweight version of the blockchain. This is the best approach for a totally decentralized PKI which does not have any additional trust requirements.

Adoption of the blockchain based approaches also introduces some issues. The major issue is the ever increasing size of the blockchain. A public blockchain can be used by any party to store information in addition to the cryptocurrency transactions that are generally stored on it. This increases the size of the blockchain, which has to be hosted by the peers in the blockchain network. This makes it difficult for smaller devices to allocate storage space for storing a copy of the blockchain. Also, blockchain requires a steady pool of miners who have to perform some computationally intensive task to add a new block to the blockchain. This incurs computational resources and electricity; however this can be thought of a service provided for a sum of money.

C. Contributions

There are three main contributions of this paper. (i) The design of three blockchain-based approaches for PKI. (ii) The Implementation and benchmark these approaches on IoT devices. (iii) A comparison of the memory, storage and computation requirements of these approaches with respect to the traditional CA-based PKI.

The rest of the paper is organized as follows. We discuss related work in Section II. We describe the drawbacks of the traditional CA-based PKI specific to IoT ecosystems in Section III. We then introduce terminology and basic concepts relevant to the paper in Section IV. We detail our approach and architectural details in Section V. We then present our experimental setup and implementation in Section VI. We then compare the performance of our approaches and their memory, storage and computation requirements in Section VII. Finally, we outline conclusions and future work in Section VIII.

II. Related Work

Several approaches have been suggested and developed for blockchain-based PKI. Fromknecht et al. [5] propose Certcoin to manage identities and corresponding public keys using a decentralized blockchain built on top of Namecoin [3]. Al-Bassam [4] proposes a PKI system using a web-of-trust model and smart contract based on the Ethereum blockchain. Matsumoto et al. [9] introduce Instant Karma PKI (IKP) that uses Ethereum Smart Contracts as an alternative to Google's transparent certificates to monitor and handle CA misbehavior. Wilson et al. [10] use Bitcoin's identity-verification transactions to enable verification of PGP certificates built upon a distributed web-of-trust. Zyskind et al. [11] use Bitcoin to construct a personal data management platform focused on privacy. Ali et al. [12] introduce Blockstack, a blockchain-based naming and storage system built on top of Namecoin and Bitcoin. Leiding et al. [13] propose Authcoin as an alternative to CA-based PKI and web-of-trust. It combines a challenge-response based validation and authentication process for domains, certificates, email accounts and public keys. However, these approaches do not consider the blockchain techniques in the context of IoT devices nor provide a concrete implementation integrated with an SSL/TLS library. Our work benchmarks the certificate verification times for different blockchain techniques deployed on IoT devices and provides a proof-of-concept implementation for the OpenSSL [6] library. Won et al. [14] propose the use of Emercoin NVS for replacing a conventional CA based PKI for IoT. However, they use blockchain for basic (key:value) pair storage which can be a limitation when more complex attributes, like access-control policies or device capabilities, need to be stored on the blockchain. Our approach addresses this limitation by leveraging the smart contract functionality provided by the Ethereum blockchain. Also, previous approaches require a full blockchain node running at a trusted server that the device can query to get updated information. We remove such a requirement by using the light sync client approach provided by Ethereum that stores only a part of the blockchain and retrieves the rest of the information on demand from the blockchain network.

III. Drawbacks of CA-based PKI for IoT

In what follows we discuss the drawbacks of CA-based PKI in the context of IoT.

- **Need to trust manufacturer generated certificates.** IoT devices are generally embedded devices, which have no user-interface. So it is really hard for the end-users, who might be individuals using the device for personal use, or network administrators who are maintaining the device for an organization, to interact with the device. Also, dealing with a CA to apply for a certificate might

be too daunting for end-users. For this reason, device manufacturers generally take care of the private/public key generation, the certificate creation and signing by a CA. This exposes the device private key to the device manufacturer, who may store it for malicious uses. Such a key can be used to decrypt all the data sent and received by that device. This requires the consumers to trust the various device manufacturers. Also, once the device is sold the manufacturer has no incentive to manage the certificate. If the certificate expires or the private key is leaked somehow, it is really difficult for the consumer to reissue a certificate and get it signed by a CA.

- **High certificate signing cost.** The certificate signing cost can be high considering the huge number of IoT devices in a single ecosystem. A single certificate can range from anywhere between $100 to $1000 depending on the specific CA and the type of the certificate required. This is a big hindrance to the widespread adoption of digital certificates and consequently TLS/SSL secure communication protocols for IoT devices.
- **Slow certificate signing process.** The traditional CA certificate signing process can take several days depending on the particular CA and the type of certificate requested. The CA has to do its due diligence to verify whether the requesting party actually owns the specific entity. Lower standards of verification to reduce the time taken to issue a certificate signature can also be a problem, because a malicious party pretending to be someone else can get through a negligent verification process and be issued a signature.
- **Difficulties in maintaining root certificate lists.** It is difficult to maintain and manage the root certificate lists in IoT devices. These lists keep getting updated, with CAs being added and removed. It is very difficult to update these lists retrospectively because these devices may not have a user interface. Even if updates to the lists are possible, it is really hard to manually update all the devices with the new certificates.
- **High certificate verification time.** The certificate verification process often involves an online certificate status protocol (OCSP) check, where the verifying entity requests the current status of the certificate from the concerned CA. This can often take multiple roundtrips (in case of chained certificates) and can significantly increase the time taken by the certificate verification process.

IV. Terminology

In this section, we introduce the blockchain terminology and concepts used in our paper.

A. Blockchain

Blockchain, first introduced by Satoshi Nakamoto [15] to implement Bitcoin, is a linked-list like data structure that has a chain of blocks with each block typically containing the hash of the previous block, a time-stamp and some form of transaction data. It is an append-only data structure and a block once added cannot be modified at a later time.

Blockchain is a peer-to-peer distributed ledger and anybody can individually verify the authenticity of the transactions recorded. There is no central server maintaining records that can be compromised. In case of a public blockchain, anyone can create a public/private key pair and participate in transactions on the blockchain. The public key of a given entity becomes its address and its private key can be used to sign any transactions on the blockchain involving the entity. The nodes of a public blockchain agree on the next block to add to the blockchain by a consensus algorithm. The most common consensus algorithm is proof-of-work, in which a subset of hosts, called miners, race to complete a computationally intensive task, and whoever finishes it first can propose the next block to be added to the blockchain.

B. Emercoin Name Value Storage

Multiple cryptocurrencies have been built upon the concepts introduced by Bitcoin and have added several additional features to it. One such cryptocurrency, Emercoin [7], provides a service to store (name: value) pairs (NVS) in its blockchain, where

- **Name** is the label for the stored data, up to 512 bytes.
- **Value** is the data itself, up to a length of 20*1024 (20kb).

Each NVS pair has an owner. The owner is specified by an Emercoin address stored together with the pair. Only the owner of the address is able to modify or delete the pair, or transfer ownership to another address.

C. Ethereum and Smart Contracts

Ethereum [8], also known as programmable blockchain, builds upon the concepts introduced by Bitcoin and adds a distributed computing platform and operating system featuring a smart contract (scripting) functionality on top of it [16]. Ethereum essentially provides a decentralized Turing-complete virtual machine, called the Ethereum virtual machine (EVM). It can be used to create decentralized application (dapps), which work according to a peer-to-peer network model rather than a client-server network model.

Ethereum enables smart contracts which are basically just object-oriented code files that run on top of the blockchain. They provide all the basic object oriented programming functionalities including modularity, encapsulation, inheritance etc. Programmers can create complex data structures, mappings and public/private functions to interact using smart contracts. The smart contracts can access accounts on blockchain and interact with other smart contracts deployed on blockchain. Examples of dapps created on the Ethereum blockchain include decentralized voting, decentralized insurance contracts and news networks.

D. Ethereum - Light Sync mode

Ethereum Light Sync mode makes it feasible for smaller devices with limited hardware resources to interact with the Ethereum blockchain. Light clients rely on the Light Ethereum

subprotocol (LES) and only download a subset of the block headers in the beginning and fetch everything else on-demand from the blockchain network when required. This allows light clients to be much more efficient in terms of storage, but increases the bandwidth overhead for the devices. Everything else remains the same including the libraries and commands that communicate with Ethereum nodes. It should be noted that this relies on the blockchain network having enough full Ethereum nodes to get actual transaction and state information. The data retrieved is reliable, provided the majority of the miners correctly follow the protocol.

V. Our Approach

Our blockchain approaches can be categorized broadly into two architectures: IoT architecture with a remote blockchain node (see Figure 1) and IoT architecture with light sync mode (see Figure 2). The first architecture requires a remote trusted blockchain node which is a part of the blockchain network that keeps an updated copy of the blockchain data. Such a remote blockchain node can be maintained by the device manager or the administrator for the specific IoT ecosystem. The second architecture does not require any TTP and a light version of the blockchain node is hosted on the IoT device itself.

We now present a high level overview of the steps to be performed for using the blockchain-based PKI:

- **Step 1:** A device manager or network administrator uses open source libraries to generate a (private key:public key) pair for the device.
- **Step 2:** The public key generated in Step 1 is then used to create a public key certificate.
- **Step 3:** A new random serial ID is generated. The ID and certificate hash are stored in the blockchain, according on the specific approach.
- **Step 4:** For the actual communication the serial ID and certificate are sent to the recipient and can be used to query the blockchain. This is explained in detail in the next section.

The proposed approaches address several shortcomings of the traditional CA-based PKI as discussed in what follows:

- **No need for centralized TTP.** The first architecture requires TTPs that host blockchain nodes. The second architecture does not require any TTP.
- **Quick addition and removal of certificates.** The certificate addition process takes minutes to under an hour with the blockchain-based PKI instead of days as in the case of CA-based PKI.
- **Resilience to distributed denial of service (DDoS) attacks.** Due to the high number of peers in the blockchain network, it is really hard to bring the whole network down by targeting it with a DDoS attack, as is possible in case of CA servers.
- **No need for a separate monitoring framework for the CA behavior.** There is no need for a separate framework, like Google Certificate Transparency, for monitoring and auditing the behavior of the multiple CAs.
- **Flexibility in storage capabilities.** The blockchain approach makes it possible to store much more information than just the certificate verification data. This approach can be extended to store the device access control policies and other communication policies on blockchain.

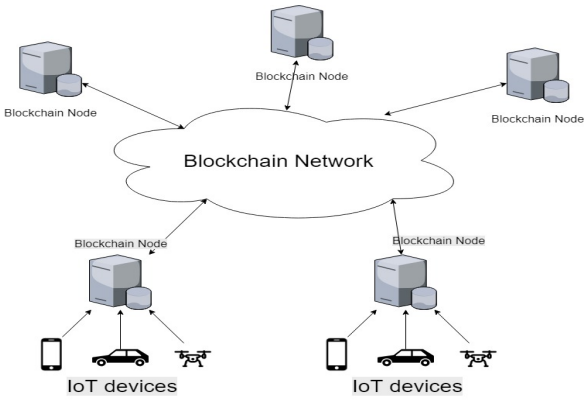

Fig. 1. Blockchain IoT architecture with Remote Node

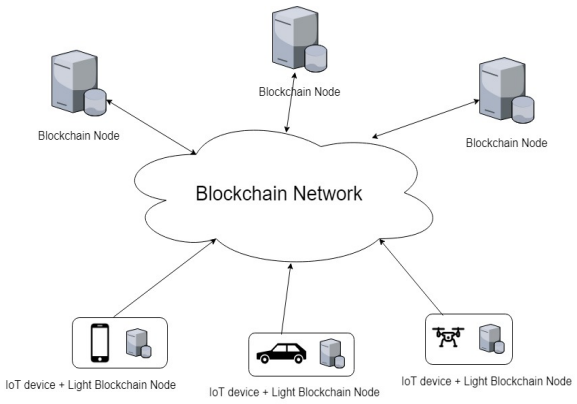

Fig. 2. Blockchain IoT architecture with Light Sync Mode

VI. Implementation

For the IoT device, we use a Raspberry Pi 2 [17] with 900 MHz 32-bit ARM Cortex-A7 and 1 GB RAM. We modified the OpenSSL [6] library (version 1.1.0) in order to support certificate verification via the blockchain approaches in addition to the traditional CA-based PKI approach. We implemented the server machine using a PC with the Intel-Core i5 6300 HQ CPU (2.30 GHz) and 8 GB RAM. The server machine is responsible for hosting the blockchain nodes in approaches 1 and 2, and answering the queries by the IoT device. Implementation details for the specific approaches are given in the following subsections.

A. Traditional CA-based PKI

For measuring the performance of the CA-based PKI, we verify the certificate of *www.google.com*, which has a 2-level

certificate chain. We use the default certificate verification functionality of OpenSSL library to verify the certificate. We add the time taken to do an OCSP check to the verification time in order to mimic a full certificate verification request.

B. Approach 1: Emercoin NVS

We create a self signed certificate using the *openssl req* command and fill in the required details for the certificate. We register the certificate on the Emercoin NVS with a 16-byte globally unique identifier (GUID) which serves as the name of the device(ID_A). The SHA-256 hash (16-bytes) of the certificate serves as the value in the NVS store. The expiration date is set on the basis of the selected time period of registration. The NVS operation details are mentioned below:

- NVS operation: $REGISTER$
- Name: ID_A
- Value: $H(Cert_A)) \rightarrow H()$ is the SHA-256 hash function.
- Expiration date: EX_A

To verify a certificate with this approach, we modify the certificate verification functionality of OpenSSL and issue a query to the Emercoin blockchain instance hosted in a server in the same local area network. This server uses the official Emercoin wallet implementation [18]. This will be the most common scenario where a network administrator will be hosting the blockchain node on the same network as the IoT devices performing the verification. The query sends the ID of the certificate and receives the hash of the certificate back. The hash is then compared with the calculated hash of the certificate received for verification. If the hashes match, the verification is successful.

C. Approach 2: Ethereum with Remote Node

We work with the Rinkeby test network for Ethereum blockchain approach. The OpenSSL verification code is given the IP address and the port of our host server running the Ethereum node. This server uses the official Ethereum wallet implementation [19] with fast sync mode. We deploy a smart contract with a hashmap of "DeviceID" and "Device Details" which includes {the owner of the device, the cryptographic hash of the certificate and whether the certificate is valid}. We reserve 16-bytes for storing the DeviceID and 32-bytes for the hash of the certificate. The contract has three public functions:

- *AddDevice(DeviceID, Hash):* To add a new device if DeviceID does not exist. It adds the device with the DeviceID and Hash of the certificate. The owner is set to be the sender of the current transaction.
- *RemoveDevice(DeviceID):* To remove a device from the storage with the DeviceID if the sender is the owner of the device, otherwise returns without doing anything.
- *GetDeviceHash(DeviceID):* To return the device Hash, given a DeviceID. Any party can call this function, not just the owner of the device.

These functions can be called from any program using an HTTP POST request to a trusted blockchain node, with the following JSON fields.

- "from": <*sender_address*>,
- "to": <*smart_contract_address*>,
- "gas": <*computational_complexity*>,
- "gasPrice": <*price_per_unit*>,
- "data": <*function_signature_with_arguments*>

The functions return the requested values and can be utilized to check whether the certificate is valid or not. GetDeviceHash function is most commonly used to get the hash of a certificate which can then be compared with the calculated hash of the certificate received for verification.

D. Approach 3: Ethereum with Light Sync Mode

The implementation details of Ethereum with Light Sync Mode are exactly the same as Approach 2, with the difference that there is no remote node that needs to be configured. A lightweight Ethereum node [19] runs on the device itself and can be used to query the Ethereum Smart Contract deployed on the blockchain.

VII. Performance Analysis and Comparison

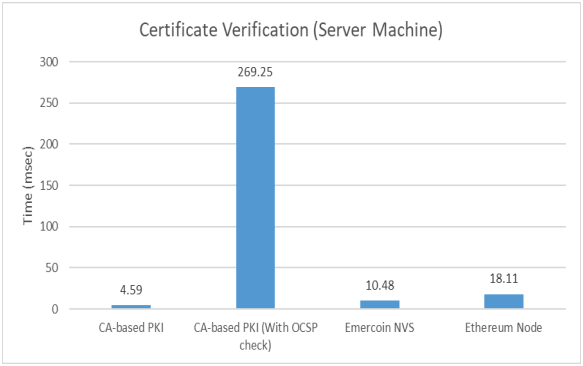

Fig. 3. Execution times on the server machine

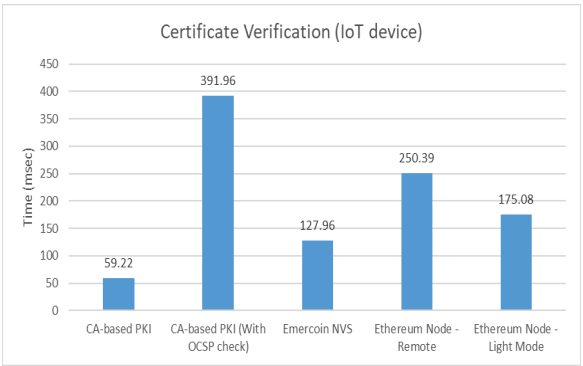

Fig. 4. Execution times on the IoT device

We plot the experimental results in Figures 3 and 4 for the server and IoT, respectively. Table I presents the comparison of the various PKI approaches.

TABLE I
COMPARISON OF PKI APPROACHES FOR IOT

Category	CA-based PKI	Emercoin NVS	Ethereum - Remote Node	Ethereum - Light Sync Mode
Time taken (msec)	59.22 (without OCSP check) and 391.96 (with OCSP check)	127.96	250.39	175.08
Storage required	Depends on the Certificate Revocation List size	0 MB	0 MB	382 MB
Trust requirement	Need to trust CA	Need to trust Remote Node	Need to trust Remote Node	No trust requirement
Cost	100s of dollars depending on CA	0.021 Emercoin ($0.07)	0.00023 Ether ($0.18)	0.00023 Ether ($0.18)
Time to issue a certificate	Multiple days	10 minutes	Less than 1 minute	Less than 1 minute

A. Execution time

Our experiments show that on the server machine, the CA-based PKI performs the best with 4.59 msec per verification, but adding the OCSP check overhead increases the verification time by a factor of x58. The reason is the network latency. Such a latency is due to DNS queries for the OCSP server, and the round-trip for the actual OCSP query to the CA server. Emercoin NVS approach outperforms the CA-based PKI verification approach with OCSP check by x25 per verification. Similarly, the Ethereum Node verification outperforms it by x14 per verification for the Rinkeby Test Network in "Fast" Sync mode. The verification time for Ethereum Rinkeby Test Network for "Light" Sync mode is also comparable at an x14 speedup.

On the IoT device, the CA-based PKI approach takes 59.22 msec for verification without the OCSP check and is x6 times slower with the OCSP check. The reason is again network latency for the OSCP query. The Emercoin NVS approach outperforms the traditional CA-based PKI approach with OCSP check by x3, the Ethereum approach with a remote node outperforms it by x1.5, whereas the Ethereum approach with "Light" Sync mode outperforms it by x2.4.

B. Storage space

The storage space required for traditional CA-based PKI might vary depending on whether the software or browser being used maintains a certificate revocation list (CRL) or not. Every CA maintains its own CRL list with different sizes. For Emercoin NVS based approach, the size of the blockchain is 240 MB at the time of the writing of this paper. For Ethereum, the blockchain size is much larger at 85 GB for "Fast" Sync mode and 382 MB for the "Light" Sync mode.

On the IoT device, the Emercoin and Ethereum remote node approach takes no space, whereas the Ethereum "Light" Sync approach takes 382 MB, which is identical to the server machine. This is reasonable for most IoT devices that have an SD memory card slot. The Raspberry Pi device we use has an 8 GB SD card (storage capacity of approximately 6 GB after OS installation).

C. Trust requirement

For CA-based PKI, the CAs have to be trusted for not being malicious or negligent in verifying and signing certificates. They also have to be trusted to maintain an updated list of revoked certificates. In case of the blockchain-based approaches there is no requirement of trust on a central third party. However, for IoT devices the remote nodes hosting the blockchain have to be trusted to provide the correct data. In case of Ethereum "Light" Sync mode, however, there is no need to place trust in any third party.

D. Cost of adding a new certificate

The cost of getting a certificate issued by a traditional CA may range from $100 to $1000 dollars depending on the type of certificate requested. Emercoin, at the time of the writing of this paper requires 0.021 EMC for storing a data of 100 bytes for 1 year, which translates to 7 cents. Ethereum on the other hand requires 0.00476 Ether for the actual contract creation, which translates to $3.68 according to the current price of Ether. Adding an entry into the Smart Contract on the other hand costs 0.00023 Ether which is approximately equal to 18 cents.

E. Storage flexibility

A CA-based PKI allows parties to store basic information in x509 certificate format. However, such a certificate only stores limited information. For example, it does not have room for storing relationships between different entities. Instead, blockchain-based PKI allows one to store additional information. For example, in Emercoin and Ethereum, the owner address of the device is also saved in the blockchain. This owner address can be used for a reverse look-up, to see how many devices an owner has under its control. Ethereum is even more flexible as it allows one to create and store more complicated data structures using smart contracts. Examples include lists and hashmaps of devices and their relationship to each other.

F. Time to issue a certificate

For traditional CA-based PKI, it may take anywhere from multiple hours to several days to issue a new certificate depending on the type of certificate requested and the specific CA signing the certificate. For Emercoin, a new block is added approximately every 10 minutes so chances are that the newly added (name:value) pair will be synced to the blockchain in an average of 10 minutes. For Ethereum, a new block is added approximately every 15 seconds, but due to the high number of outstanding transactions waiting for confirmation,

a new request might take more than that. Right now, it takes anywhere from 48 seconds to 30 minutes for confirmation of a new transaction. This process can be sped up depending on the urgency, by offering more fees to miners for prioritizing a transaction.

VIII. Conclusion and Future Work

In this paper, we propose and evaluate three different blockchain-based alternatives to traditional CA-based PKI for certificate management. We prove their feasibility in the context of the resource constrained IoT devices in terms of computational power, memory and storage capabilities. In the paper, we also discuss how these approaches address the shortcomings of CA-based PKI while maintaining or improving the certificate verification performance and removing the requirement of multiple TTPs.

We plan to extend the Ethereum smart contracts approach to be able to store and retrieve more information about the IoT devices like access control policies and operating parameters. A hierarchical organization of the (name:value) storage will be also investigated. In addition, we plan to implement and benchmark our own public blockchain, which can be used for our specific purposes instead of serving as an all-purpose store.

IX. Acknowledgements

The work reported in this paper is partially supported by the NSF grant CNS-1719369 and Intel as part of the NSF/Intel ICN-WEN program.

References

[1] IDC, "Connecting the iot," 2015. [Online]. Available: http://www.idc.com/infographics/IoT/ATTACHMENTS/IoT.pdf
[2] D. Cooper, "Internet x. 509 public key infrastructure certificate and certificate revocation list (crl) profile," 2008.
[3] Namecoin, "Namecoin," 2017. [Online]. Available: https://namecoin.org
[4] M. Al-Bassam, "Scpki: a smart contract-based pki and identity system," in *Proceedings of the ACM Workshop on Blockchain, Cryptocurrencies and Contracts*. ACM, 2017, pp. 35–40.
[5] C. Fromknecht, D. Velicanu, and S. Yakoubov, "A decentralized public key infrastructure with identity retention." *IACR Cryptology ePrint Archive*, vol. 2014, p. 803, 2014.
[6] OpenSSL Software Foundation, "Openssl," 2017. [Online]. Available: https://www.openssl.org
[7] Emercoin, "Emercoin," 2017. [Online]. Available: http://emercoin.com
[8] Ethereum, "Ethereum," 2017. [Online]. Available: https://ethereum.org
[9] S. Matsumoto and R. M. Reischuk, "Ikp: Turning a pki around with blockchains." *IACR Cryptology ePrint Archive*, vol. 2016, p. 1018, 2016.
[10] D. Wilson and G. Ateniese, "From pretty good to great: Enhancing pgp using bitcoin and the blockchain," in *International Conference on Network and System Security*. Springer, 2015, pp. 368–375.
[11] G. Zyskind, O. Nathan, and A. Pentland, "Decentralizing privacy: Using blockchain to protect personal data," in *Security and Privacy Workshops (SPW), 2015 IEEE*. IEEE, 2015, pp. 180–184.
[12] M. Ali, J. C. Nelson, R. Shea, and M. J. Freedman, "Blockstack: A global naming and storage system secured by blockchains." in *USENIX Annual Technical Conference*, 2016, pp. 181–194.
[13] B. Leiding, C. H. Cap, T. Mundt, and S. Rashidibajgan, "Authcoin: validation and authentication in decentralized networks," *arXiv preprint arXiv:1609.04955*, 2016.
[14] J. Won, A. Singla, E. Bertino, and G. Bollella, "Decentralized public key infrastructure for internet-of-things," in *MILCOM*, 2018 (Accepted).
[15] S. Nakamoto, "Bitcoin: A peer-to-peer electronic cash system," 2008.
[16] J. Dienelt, "Understanding ethereum," *New York, NY: CoinDesk*, 2016.
[17] Raspberry Pi 2, "https://www.raspberrypi.org/," 2015.
[18] Emercoin, "Emercoin wallet," 2017. [Online]. Available: https://github.com/emercoin/emercoin
[19] Ethereum, "Ethereum wallet," 2017. [Online]. Available: https://github.com/ethereum/mist