EVOKE: Efficient Revocation of Verifiable Credentials in IoT Networks

Carlo Mazzocca, *University of Bologna;* Abbas Acar and Selcuk Uluagac, *Cyber-Physical Systems Security Lab, Florida International University;* Rebecca Montanari, *University of Bologna*

https://www.usenix.org/conference/usenixsecurity24/presentation/mazzocca

This paper is included in the Proceedings of the 33rd USENIX Security Symposium.

August 14–16, 2024 • Philadelphia, PA, USA

978-1-939133-44-1

Open access to the Proceedings of the 33rd USENIX Security Symposium is sponsored by USENIX.

EVOKE: Efficient Revocation of Verifiable Credentials in IoT Networks

Carlo Mazzocca[1], Abbas Acar[2], Selcuk Uluagac[2], Rebecca Montanari[1]

[1]University of Bologna, Bologna, Italy
[2]Cyber-Physical Systems Security Lab, Florida International University, Miami, Florida, USA
{carlo.mazzocca, rebecca.montanari}@unibo.it, {aacar001, suluagac}@fiu.edu

Abstract

The lack of trust is one of the major factors that hinder collaboration among Internet of Things (IoT) devices and harness the usage of the vast amount of data generated. Traditional methods rely on Public Key Infrastructure (PKI), managed by centralized certification authorities (CAs), which suffer from scalability issues, single points of failure, and limited interoperability. To address these concerns, Decentralized Identifiers (DIDs) and Verifiable Credentials (VCs) have been proposed by the World Wide Web Consortium (W3C) and the European Union as viable solutions for promoting decentralization and "electronic IDentification, Authentication, and trust Services" (eIDAS). Nevertheless, at the state-of-the-art, there are no efficient revocation mechanisms for VCs specifically tailored for IoT devices, which are characterized by limited connectivity, storage, and computational power.

This paper presents EVOKE, an efficient revocation mechanism of VCs in IoT networks. EVOKE leverages an ECC-based accumulator to manage VCs with minimal computing and storage overhead while offering additional features like *mass* and *offline revocation*. We designed, implemented, and evaluated a prototype of EVOKE across various deployment scenarios. Our experiments on commodity IoT devices demonstrate that each device only requires minimal storage (i.e., approximately 1.5 KB) to maintain verification information, and most notably half the storage required by the most efficient PKI certificates. Moreover, our experiments on hybrid networks, representing typical IoT protocols (e.g., Zigbee), also show minimal latency in the order of milliseconds. Finally, our large-scale analysis demonstrates that even when 50% of devices missed updates, approximately 96% of devices in the entire network were updated within the first hour, proving the scalability of EVOKE in offline updates.

1 Introduction

Today, there are 15 billion Internet of Things (IoT) devices connected worldwide [46], generating an enormous amount of data that paves the way for novel applications. In such settings, the lack of *trust* [35] is one of the major concerns that limit the full usage of data and collaboration among different entities. Third-party organizations do not rely on external data for decision-making and devices do not trust each other. Therefore, increasing trust in IoT networks would remarkably contribute to leveraging a larger amount of data, leading to the development of next-generation services [43].

Centralized Approaches. At the state-of-the-art, Public Key Infrastructure (PKI) is the most widespread technology adopted to identify devices and establish mutual trust [22, 23]. In this, digital identities are traditionally issued and managed by centralized certification authorities (CAs). However, centralized approaches present several concerns and limitations, including scalability, lack of control over own information, single point of failure, and limited interoperability.

Decentralized Approaches. To address these concerns, decentralized approaches are increasingly attracting the interest of the industry and academia. In this direction, the World Wide Web Consortium (W3C) has formalized and standardized Decentralized Identifiers (DIDs) [54] and Verifiable Credentials (VCs) [55] as viable solutions for promoting decentralized identification. A DID is a unique identifier that resolves to a DID Document containing public keys, enabling the proof of ownership over specific data. On the other hand, a VC is a statement about an entity that can be cryptographically verified by a third party. VCs offer an alternative to traditional digital certificates for establishing trust and verifying the authenticity of attributes. Indeed, interest in DIDs and VCs has seen remarkable growth in recent years as they have laid the groundwork for innovative identity solutions like digital healthcare passports [15]. Such a trend can also be observed in several initiatives that are emerging worldwide with many countries that are starting to use these technologies [26, 40, 51].

For instance, VCs are considered a key technology by the European regulation for "electronic IDentification, Authentication, and trust Services" (eIDAS) to ensure secure and remote identity proofing [38]. Furthermore, DIDs and VCs

have the potential to efficiently implement the concept of Identity of Things (IDoT) [25] introduced by the Kantara Initiative. Each device is assigned a DID that enables proving certain capabilities, through VCs, to other devices.

Problem Definition. A critical operation for the trust of the whole network depends on the efficient revocation of VCs for malfunctioning or compromised devices. A notable example is the chip vulnerability on RSA key generation, as highlighted in CCS'17 by Namec et al. [37], which resulted in the revocation of more than 700K certificates for devices using that chip. In IoT networks, the presence of heterogeneous devices, some with limited computing and storage capabilities, adds complexity to the revocation process. Additionally, an IoT setting can be constrained as well and characterized by limited bandwidth, low transmission range, dynamic topology, and unreliable connectivity. These factors pose challenges to the adoption of traditional revocation mechanisms like the Online Certificate Status Protocol (OCSP) [42] and Certificate Revocation Lists (CRLs) [10], which respectively assume reliable connections and large amounts of memory. Despite the growing interest in DID and VCs [20, 33], the development of efficient approaches for VC revocation remains in its early stages as evidenced by the lack of a dedicated W3C standard [53].

EVOKE. Motivated by the growing interest in DID and VCs, coupled with the inherent challenges posed by IoT devices, in this paper, we introduce EVOKE, a novel solution to address the efficient VC revocation in IoT settings. Our approach allows devices to establish trusted interactions while minimizing computing and storing overhead. To achieve this, EVOKE leverages an Elliptic Curve Cryptography (ECC)-based accumulator [52], which efficiently aggregates multiple data into a unique element called accumulator value, whose size remains constant regardless of the number of accumulated elements. An additional significant characteristic is that proving membership in the accumulator value also demands constant verification time. These characteristics ensure that EVOKE maintains efficiency even when the number of credentials and devices within the network grows.

Implementation and Evaluation of EVOKE. We implemented and evaluated a prototype of EVOKE, assessing its effectiveness and applicability across various scenarios. First, to validate its feasibility in real-world environments, we evaluated EVOKE on commodity IoT devices. To the best of our knowledge, it is the first tool to enable using VCs on IoT devices. However, due to the restrictions imposed by vendors, as end-users, we were not allowed to run our code directly on them. Thus, to broaden the scope of our analysis and encompass a wider range of scenarios, we extended our evaluation to include hybrid networks that combine both IoT devices and Raspberry Pis. This setup allowed us to create the typical IoT deployment scenarios like star or mesh topologies that require the IoT devices' interaction with each other. Last but not least, we conducted scalability tests by simulating a large-scale network consisting of thousands of devices.

The experimental results demonstrate that EVOKE can be supported by off-the-shelf IoT devices easily, demanding only minimal requirements in terms of storage and computing resources. Remarkably, each device only needs to allocate approximately 1.5 KB of space for storing verification information, i.e., the accumulator value and proof of inclusion (witness). The constant size of the accumulator value, combined with the low latency introduced by the verification operations, significantly enhances the scalability of EVOKE and it enables *mass revocation*. Our mechanism also offers *offline revocation*, whereby devices that fail to receive revocation updates can still be updated by trustworthy devices that have received the updates. Even in the extreme case where 50% of devices miss updates in a network of one million devices, EVOKE is capable of updating almost the entire network through a limited number of interactions.

Contributions. The main contributions of the paper can be summarized as follows:

- We present a novel and efficient revocation mechanism for VCs in IoT networks. Our approach offers a direct application to device-to-device trust establishment, significantly enhancing revocation efficiency while enabling the capability for mass revocation of VCs.
- EVOKE supports offline revocation and effectively reduces network overhead. In cases where a device misses updates (i.e., accumulator value and witness), EVOKE allows for the updating of the device by leveraging interactions with other devices in the network.
- EVOKE ensures that each device stores only a minimal amount of data necessary for validating other VCs and establishing trusted communications. This design minimizes storage requirements while maintaining the integrity of the system.
- To comprehensively evaluate the effectiveness and performance of our approach, we conducted a series of experiments that include direct evaluations of EVOKE also on several commodity IoT devices. To the best of our knowledge, this work represents the first solution and design of a VC revocation scheme specifically tailored for IoT technologies.

Organization. The remainder of this paper is structured as follows: Section 2 provides the background for DID, VCs, and cryptographic accumulators. Section 3 presents the EVOKE system model and threats. Section 4 describes the revocation mechanism. Section 5 analyzes the security of our proposal. Section 6 evaluates our revocation mechanism and presents experimental results. Section 7 compares EVOKE to other approaches. Section 8 covers related work, and Section 9 concludes the paper.

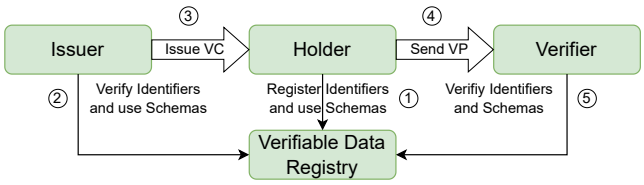


Figure 1: Overview of VC main actors.

2 Background

In this section, we provide the needed background to understand the data structure that enables revoking VCs in our proposal.

2.1 Decentralized Identifiers (DIDs) and Verifiable Credentials (VCs)

DIDs have revolutionized identification in decentralized identity frameworks [34]. A DID uniquely identifies a DID Subject, whether human or non-human entities. It comprises three elements: the Uniform Resource Identifier (URI), the specific DID method identifier, and the method-specific DID identifier. Additionally, a DID URL extends the basic DID by incorporating additional URI components for precise resource location. Each DID resolves to a machine-readable JSON-LD document called DID Document, containing cryptographic public keys, service endpoints, authentication parameters, timestamps, and metadata. DIDs eliminate the need for identity providers and centralized authorities, allowing entities to prove ownership through private keys corresponding to the embedded public keys in the DID Document. Verification is facilitated by accessing the public DID Document, shared via a verifiable data registry usually implemented using distributed ledger technologies (DLTs).

VCs are another W3C specification, providing an interoperable data structure for representing cryptographically verifiable and tamper-proof claims. VCs play a significant role in an ecosystem that includes holders, issuers, verifiers, and a verifiable data registry. Holders are entities with control over one or more VCs, while issuers create new VCs. In an IoT environment, issuers may be the manufacturer of devices. Verifiers obtain VCs for establishing trust, e.g., smart locks requesting trusted data from smoke detectors to facilitate building evacuation. The verifiable data registry mediates the creation and verification of identifiers, keys, credential schemas, and other relevant data. A VC comprises elements such as the subject URI, the issuer's URI, unique credential identifiers, claim expiration conditions, and cryptographic signatures. Moreover, the W3C VC Working Group introduced Verifiable Presentations (VPs), defining methods for signing and presenting VCs by holders. Figure 1 sketches the main actors within the ecosystem of VCs.

2.2 Cryptographic Accumulators

The academic interest in cryptographic accumulators dates back to 1994 when they were initially proposed as solutions for eliminating trusted central authorities [3]. Accumulators are used to cryptographically produce a short binding commitment to a set of elements called accumulator value, proving the membership of data in the dataset through short proofs. An accumulator is defined as a family of *one-way hash functions* that satisfy the capability of being *quasi-commutative*. A one-way hash function h is a function that takes an input value v of arbitrary length and produces a fixed-size output named hash value $h(v)$. The one-way property of the hash function means that given $h(v)$, it is computationally infeasible to recover the original input. The quasi-commutative property makes the accumulator independent from the order of values accumulated. These properties allow accumulators to be used for condensed representing a set of elements and efficiently verify membership. To test if a v_i is included in the accumulator value, one needs to calculate its corresponding witness w_i, which is obtained by accumulating all the values except v_i. Accumulators that enable verifying the inclusion of elements via *membership witnesses* are called *positive*, while those that allow verifying if elements are not accumulated through *non-membership witnesses* are referred to as *negative*. Accumulators that support both functionalities are known as *universal*.

Different types of accumulators have been proposed over the years. Extensive studies [29, 50, 56] have demonstrated that ECC-based accumulators outperform alternative solutions such as RSA-based accumulators and Merkle Trees (MTs) [47]. ECC-based accumulators offer several advantages, including smaller proofs of membership (witnesses) and efficient verification, thanks to the reduced number of mathematical operations needed.

In this paper, we leverage the features of a positive ECC-based accumulator [52] that supports batch operations, including witness generation. In contrast, the size of the accumulator value in an RSA-based accumulator is determined by the size of the modulus used in the RSA encryption scheme. The modulus size is directly related to the security level of the RSA accumulator. Thus, RSA demands larger module sizes to achieve the same security level as ECC. In MTs, the proof size increases logarithmically with the size of the committed data, which can make it grow considerably when considering a large set of data. This can be a disadvantage regarding storage space and computation time when dealing with large amounts of data.

3 System and Threat Model

In this paper, we consider IoT networks $\mathcal{N}$ that comprise issuers I, accumulators $\mathcal{ACC}$, a distributed ledger b, and IoT devices $\mathcal{D}$. In real-world scenarios, devices interact within

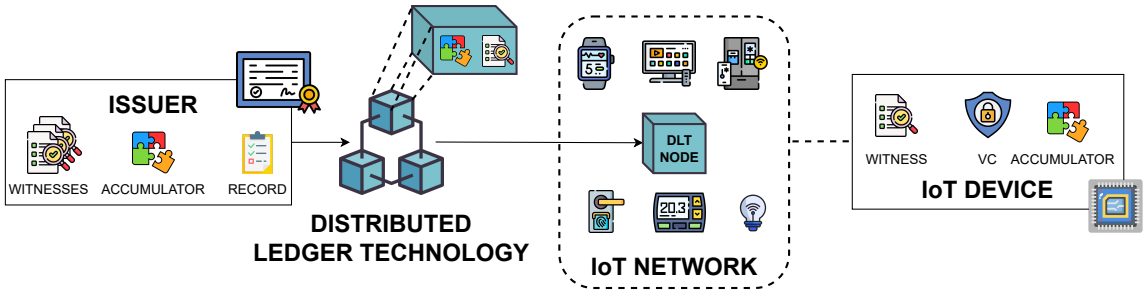


Figure 2: EVOKE architecture. The dotted lines represent a closer examination of revocation information shared on the DLT and data maintained by each IoT device.

Table 1: EVOKE Notation.

Symbol	Description
$\mathcal{N}$	Set of IoT Networks
I	Set of issuers
$\mathcal{ACC}$	Set of accumulators
j	Geographical area
A_j	Single accumulator
a_j	Single accumulator value
i_j	Single issuer
D_j	Set of IoT devices in j
d_i	Single IoT device
W_j	Set of witnesses in j
w_i	Witness of device d_i
b	DLT

specific geographical areas, whose boundary depends on several factors such as administrative divisions, technological infrastructure, or network coverage (e.g., Wi-Fi networks). For instance, a geographical area can be mapped into a 5th-generation network cell.

Without loss of generality, we assume that a geographical location j represents an IoT network $n_j \in \mathcal{N}$. Each IoT device $d_i \in D_j \subset \mathcal{D}$ in n_j has a VC issued by the issuer $i_j \in I$. To manage VCs, i_j accumulates all the valid VCs of j, i_j accumulates them in $a_j \in A_j \in \mathcal{ACC}$. Table 1 summarizes the notation used in the EVOKE system model and security analysis. In the design of EVOKE, we make the following assumptions:

1. The issuer i_j possesses sufficient computing capabilities, including an adequate level of processing power, memory, and necessary software/tools and mechanisms, to successfully build the accumulator value a_j, remove VCs, generate witnesses W_j for all $d_i \in D_j$, and disseminate updates (i.e., accumulator and witnesses) through the DLT. i_j provides VCs only to authenticated and properly functioning devices that have not been compromised.
2. Each device d_i is initially provided with a valid VC, a_j, w_i, its cryptographic material and the public key of i_j, which has issued the VC. Through the public key of i_j, d_k can establish secure and authenticated communications.
3. There are devices in n_j that have full connectivity, intended as a reliable connection to the DLT to receive the updates. Updated revocation information is disseminated in n_j through a publish/subscribe paradigm.

3.1 Components

In this section, we provide a detailed explanation of EVOKE's components, whose architecture is depicted in Figure 2.

- *Issuer*: The issuer i_j is responsible for issuing and revoking VCs. It records all issued VCs and IoT devices within the network. The issuer operates independently from the rest of the network and only communicates through the DLT. We assume it possesses sufficient computing capabilities for managing VCs, including the generation and update of the accumulator value a_j and witnesses W_j.
- *DLT*: b serves as a secure, distributed data source for exchanging updates (i.e., a_j and W_j), between the i_j and D_j. DLTs are secure due to the use of cryptographic techniques like digital signatures and hash functions, along with the resilience to single points of failure and the adoption of consensus mechanisms. In the highly improbable case of being compromised, the availability of updates may be affected, but the updates' integrity remains intact. Updates are included in VCs signed by i_j, thus only the issuer can generate them. Each n_j has at least one DLT node equipped with adequate resources to maintain a consistent and up-to-date version of b. In our work, we utilized a directed acrylic graph (DAG)-based ledger, DAG for brevity. The adoption of a DAG offers transparency and immutability of shared data while exhibiting enhanced performance in terms of latency and energy consumption that make it ideal for IoT environments [17].
- *IoT Device*: $d_i \in D_j$ vary in computational power, resource consumption, storage capacity, and networking capabilities. Each IoT device d_i is equipped with sufficient storage for essential components such as the a_j,

d_i, a VC, DID, and associated information. Additionally, each d_i possesses the minimum cryptographic functions necessary to establish trust within the IoT network n_j. Devices communicate with each other through their networking technologies, while only devices with sufficient networking capabilities can interact with b.

3.2 Threat Model

The security of our scheme relies heavily on the secure implementation of the proposed system. We assume that the a_j has been computed securely, ensuring its collision resistance. To assess the security of our system, we employ the Dolev-Yao model adversary [14], who possesses the capability to eavesdrop on, intercept, or inject an arbitrary number of messages. Furthermore, we also consider that the attacker can compromise the entities composing n_j. The primary objective of the adversary is to acquire a valid VC, thereby gaining the ability to establish trusted communications. This goal can be accomplished by compromising i_j, D_j, or communication links. In the following, we highlight the primary threats of a scenario where an issuer furnishes VCs for IoT devices:

- **Compromised Issuers**: Gaining control over the issuer i_j gives the adversary the capability to arbitrarily issue valid VCs as well as revoke valid ones. The attacker can generate VCs with false or misleading claims, improperly granting permission to entities within the system. Furthermore, the adversary may invalidate legitimately issued VCs, resulting in a denial of services (DoS) or disruption of the established trust relationship.
- **Compromised IoT Devices**: IoT devices D_j act as the entry point to IoT networks and can be vulnerable to a range of attacks if compromised by adversaries. The attacker may identify vulnerabilities in the firmware of IoT devices to gain unauthorized access or control over other devices. Devices can be compromised in a variety of ways, such as malware distribution through firmware updates or the devices may have already been manufactured with compromised hardware or software components in case the adversary has direct access.
- **Compromised Communication**: In n_j, D_j are not obligated to implement encryption mechanisms, making it relatively easy for the adversary to eavesdrop on the traffic. For example, the attacker may use the intercepted VC to spoof the identity of the device and establish trust on its behalf.

4 EVOKE Protocol

The proposed approach allows IoT devices D_j to securely communicate with minimal overhead. Specifically, each device d_i maintains the accumulator value a_j, which comprises all the valid VCs, and a witness w_i that demonstrates the validity of its credential. a_j and the witnesses W_j are computed by the issuer i_j and initially provided to all $d_i \in D_j$. However, when a VC is revoked both a_j and W_j have to be updated. Using just two values, which require only a few bytes, remarkably reduces the revocation management overhead both in terms of storage and distribution. The remainder of this section provides the full details of our method.

4.1 Preliminaries

The accumulator value a_j is the key element of our proposal that enables efficient management of VCs. In this subsection, we provide all the details related to its usage in EVOKE.

Proof of membership: When using a revocation list, two devices d_i and d_k that have to authenticate each other present their VCs and verify whether the received one is included in the list. Similarly, to use a_j to revoke VCs, d_i needs to present proof of membership named witness. In this way, the receiver applies a verification function that allows it to state whether the VC has been included in a_j.

Inputs of the accumulator: Values in a_j are derived from the hash conversion of the valid VCs. Therefore, the security of a_j relies heavily on the chosen hash function. In our case, we use the widely adopted SHA-256 hash function, which is considered to be secure and collision-resistant.

Functions: The following functions are used to implement the proposed approach:

- $a_{info} \leftarrow \mathsf{Setup}(\mathsf{K})$: setups the parameters of $\mathsf{a_j}$ by taking a set of public values K, which represent the random generators of the elliptic curve.
- $a_j \leftarrow \mathsf{ComputeAccumulator}(\mathsf{a_{info}}, \mathsf{V})$: accumulates a set of valid verifiable credentials V by mapping them onto the curve. This function is also executed to update a_j when V is modified, as VCs can be revoked and newly valid ones can be issued.
- $a_j^t \leftarrow \mathsf{RevomeFromAccumulator}(\mathsf{a_{info}}, \mathsf{a_j^{t-1}}, \mathsf{V})$: removes a set of invalid verifiable credentials V. This function is also executed to update a_j.
- $w \leftarrow \mathsf{ComputeWitness}(\mathsf{a_j}, \mathsf{vc_i})$: generates a witness value $\mathsf{w_i}$ for a valid verifiable credential $\mathsf{vc_i}$, providing proof of its membership within $\mathsf{a_j}$.
- $0,1 \leftarrow \mathsf{Verify}(\mathsf{a_j}, \mathsf{vc_i}, \mathsf{w_i}, \mathsf{pk_i})$: $\mathsf{d_i}$ presents $\mathsf{vc_i}$ by signing it with its secret key $\mathsf{sk_i}$. The verifier executes this function to authenticate $\mathsf{d_i}$. It verifies whether $\mathsf{w_i}$ provided by $\mathsf{d_i}$ is accumulated in $\mathsf{a_j}$. The verifier uses the device's public key $\mathsf{pk_i}$ to ensure that $\mathsf{vc_i}$ is associated with $\mathsf{d_i}$.
- $a_j^t \leftarrow \mathsf{UpdateAccumulator}(\mathsf{a_{info}}, \mathsf{a_j^{t-1}}, \mathsf{V})$: removes a set of invalid verifiable credentials V through the $\mathsf{RevomeFromAccumulator}()$ function. In case, a new set of valid verifiable credentials V is released, they are included in a_j using the $\mathsf{ComputeAccumulator}()$ function.
- $w^t \leftarrow \mathsf{UpdateWitness}(\mathsf{a_j^t}, \mathsf{vc_i})$: updates the witness corresponding to a valid verifiable credential $\mathsf{vc_i}$. When $\mathsf{a_j}$ is

updated, previously disseminated W_j^{t-1} become invalid. This function takes the updated value a_j^t, a verifiable credential vc_i, and returns a valid witness w_i^t corresponding to vc_i.

4.2 Protocol

This subsection describes the main phases of our protocol, which uses the features of the positive accumulator presented in [52]. These capabilities are leveraged by EVOKE to verify whether a VC has been revoked or not, i.e., checking if the corresponding witness is included in the accumulator value. Given two IoT devices d_i and d_k, they can validate each other as long as the accumulator value a_j and their respective witnesses, w_i and w_k, are updated. To securely share a_j among all $d_i \in D_j$, i_j includes the accumulator value a_j, as well as the witnesses W_j, in self-signed VCs and publishes them on b. This approach enables D_j to verify the authenticity before updating. Furthermore, if d_i is outdated and cannot directly download the freshly issued a_j and w_i, the updated d_k can share the refreshed VCs needed to establish mutual trust. The proposed protocol supports mass and offline revocation, which are defined as follows:

Definition 1 (Mass Revocation) *Mass revocation is the capability of efficiently revoking a large number of VCs without impacting the issuer and device overhead. A protocol designed for mass revocation ensures that the memory size of the underlying data structure to manage VCs as well as the verification time remain unaffected by the number of revoked VCs.*

Definition 2 (Offline Revocation) *Offline revocation refers to the capability of devices to receive and apply updates even when they have been disconnected from the network, without burden on network resources. A protocol that supports offline revocation is designed to minimize the amount of data transmission overhead required to facilitate these updates.*

Setup. In the setup phase, i_j securely generates the accumulator's parameter K and initializes a_j by executing the Setup() function. Valid VCs are accumulated into a_j through the ComputeAccumulator() function. For each valid VC, i_j generates a hash value, which serves as a unique identifier, which is then used to obtain the corresponding representation on the curve. After computing the final a_j, i_j generates the witnesses $w_i \in W_j$ for each valid VC by using the ComputeWitness() function. W_j provides cryptographic proof of the inclusion of the VC in a_j. Initially, each d_i receives a_j, a valid VC, and its corresponding w_i. These components are essential for subsequent operations.

Authentication. During the authentication, d_i initiates the communication with d_k by sending the timestamp of a_j. The issuer associates a timestamp with the accumulator value that

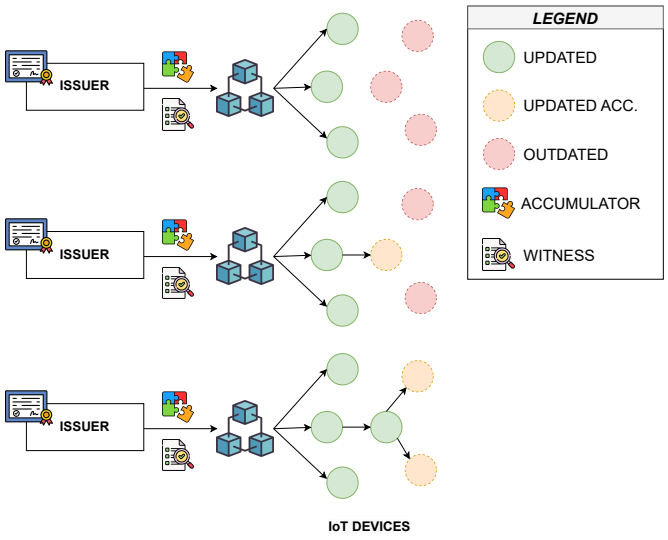


Figure 3: Updating accumulator value and witnesses.

serves as a means to synchronize the devices and ensure consistency. Device d_k compares the received timestamp with that of its accumulator value. Matching timestamps indicate that both d_i and d_k are operating with the same version of a_j, and they can proceed with the authentication. However, if the timestamps differ, it implies that one of the devices has an outdated a_j. In this case, the device with the older timestamp updates its a_j to the latest version.

Once synchronized, d_i and d_k exchange signed VCs and the corresponding witnesses, namely w_i and w_k. Both devices verify that the received VC has been issued by i_j, if the verification is successful they hash them and utilize the Verify() function to check if the VC belongs to that device and if the hash has been included in a_j.This verification step ensures that the VCs are valid and have not been tampered with. If both VCs are found within a_j, it indicates their membership and authenticity. Consequently, the devices can trust each other and establish a secure and trusted communication channel.

Update. Over time, the issuer i_j may need to revoke certain VCs due to device compromise or faults. Revoking VCs entails updating the accumulator value a_j and the corresponding witnesses W_j for the IoT devices that are still functioning correctly. Figure 3 offers a high-level overview of our protocol while updating an IoT network. To revoke VCs, i_j executes the UpdateAccumulator() function that removes invalid credentials from a_j.Once the updated a_j is obtained, i_j uses the novel version to generate a new W_j for the remaining valid VC. The UpdateWitness() function is employed to calculate the updated witnesses based on the modified a_j. Subsequently, i_j embeds the updated information, including the newly generated W_j, within VCs signed by i_j to ensure their authenticity. These modified VCs, containing the refreshed witnesses, are

then shared through a DAG b. By distributing the updated VCs with their corresponding witnesses, other devices in the network can be informed about the revoked credentials and have access to the latest valid set. This mechanism ensures that all devices have an up-to-date view of the trusted credentials within the system, promoting secure and reliable communication.

In an IoT network n_j, offline updates are essential for devices with intermittent connectivity or power-saving hibernation. EVOKE leverages VCs to share a_j and W, enabling devices to update even when not directly connected. When an updated device d_i encounters an outdated device d_k, it can facilitate the update process by sharing a VC that contains a more recent version of a_j. Depending on the networking capability of d_k, there are two possible approaches: *direct retrieval* and *indirect retrieval*. If d_k supports direct retrieval, it has sufficient networking capabilities to directly retrieve the corresponding w_k from b. Otherwise, it obtains w_k from d_i or another device with satisfactory network connectivity. This indirect retrieval process ensures that d_k receives the necessary w_k to accompany the updated a_j. Once d_k acknowledges having an outdated version of new a_j, it disables trusted communications until it will not receive updates. This precautionary measure prevents the authentication of other devices that may attempt to leverage an outdated version of a_j. Importantly, both a_j and w_k should be included within a VC to facilitate the sharing process. By utilizing VCs to share a_j and W, the EVOKE system ensures that updates are always trusted, even if not directly provided by i_j. Algorithm 1 shows how updates are handled in n_j.

Algorithm 1 `HandleUpdates`: Function executed to update $d_i \in D_j$. If d_i becomes outdated, it can still receive revocation information from updated devices.

Input: D_j, b
Output: Updated D_j
for each $d_i \in D_j$ **do**
    d_i interacts with some $d_k \in D_j$ with $i \neq k$
    if a_j^i is older than a_j^k **then**
        $a_j^i \leftarrow a_j^k$
        if d_i has sufficient connectivity **then**
            $w_i \leftarrow DirectRetrieval(b, d_i)$
        else if d_k has sufficient connectivity **then**
            $w_i \leftarrow IndirectRetrieval(b, d_k, d_i)$
        else
            disable trusted communications for d_i
        end if
    else if a_j^k is older than a_j^i **then**
        $a_j^k \leftarrow a_j^i$
        if d_k has sufficient connectivity **then**
            $w_k \leftarrow DirectRetrieval(b, d_k)$
        else if d_i has sufficient connectivity **then**
            $w_k \leftarrow IndirectRetrieval(b, d_i, d_k)$
        else
            disable trusted communications for d_k
        end if
    end if
end for

Discussion. EVOKE's ability to handle intermittent connectivity and hibernation ensures that IoT devices D_j can remain up-to-date with a minimal delay, thereby maintaining the system's integrity and security. However, when two previously offline devices, d_i and d_k, interact, it is important to consider two possible scenarios that require further attention.

- *Consistent Accumulator Value Version*: If both devices d_i and d_k share the same version of the accumulator value a_j, which happens to be outdated, they are unable to detect any missed updates. In this case, devices may incur misbehaved authentication if one of them has a revoked VC. As soon as these devices encounter a device with a fresher accumulator version and one of them supports direct retrieval, updates are applied, and their correct behavior is re-established. Let us note that this situation is unlikely to occur. Additionally, it is worth noting that this undesired condition has a time limitation. As shown in Subsection 6.4, the majority of the devices are updated within the first hour, reducing this risk.
- *Mismatching Accumulator Value Version*: When both the devices have a mismatching version of a_j, additional steps are required. Let us assume that d_k has the older version of a_j. If d_k can implement direct retrieval, it retrieves w_k and recognizes that it corresponds to a fresher version of a_j than what was exchanged. As a result, d_k obtains the associated novel version of a_j and updates itself accordingly. On the other hand, if indirect retrieval is necessary, d_i collects w_k and performs a similar verification process. Eventually, d_i acquires the latest version of a_j and the corresponding witnesses and updates d_k accordingly. If both devices lack direct retrieval, they temporarily disable their trusted communications while waiting for updates in subsequent iterations. In Subsection 6.4, we demonstrate that nearly the entire network is updated within the first hour, even if 50% of devices missed updates.. By following this procedure, devices ensure that their accumulator values and witnesses are synchronized, allowing for consistent and trusted communication.

These scenarios highlight EVOKE's adaptability, allowing D_j to reconcile differences in accumulator value versions and update each other for consistency. Through these mechanisms, EVOKE promotes seamless communication and guarantees the overall security and reliability of the IoT network n_j.

5 Security Analysis

In this section, we consider the threats introduced in Subsection 3.2 and analyze how they are mitigated in an environment where EVOKE is employed as a revocation mechanism. It is worth noting that threats linked with the issuer (e.g., secure private key storage), generally refer to any trusted entity responsible for managing VCs or certificates. Therefore, they are not directly mitigated by revocation mechanisms, including EVOKE. Additionally, revocation mechanisms are not responsible for detecting compromised devices while they have to manage their revocation upon a device being marked as compromised.

5.1 Security of Issuer

The issuer i_j is the primary target of the adversary as it manages the accumulator, which is the key data structure for handling VCs. We examine three potential attack scenarios and corresponding countermeasures:

Securing the Issuer's Environment. i_j is deployed in a secure environment that imposes restrictions on direct external communication. Valid VCs are accumulated within the issuer's secure perimeter, and a_j is shared with n_j through a DAG b. This setup makes it significantly challenging for the attacker to gain unauthorized access a_j from external sources.

Compromise of Accumulator Security Measures. Despite robust defense mechanisms in place, the adversary may still attempt to compromise the security measures protecting a_j and acquire the setting parameters $\mathsf{a}_{\mathsf{info}}$. However, even if the adversary gains access to these parameters, it does not provide them with an advantage in attacking the revocation management process. To undermine our approach, attackers would need to obtain the key pair associated with the issuer DID and use it to sign fraudulent VCs that comprise the accumulator value a_j and witnesses W_j.

Key Pair Compromise. If an adversary successfully obtains the issuer's key pair, immediate action must be taken to mitigate the risk. This includes migrating i_j to a new server and providing it with a fresh DID and key pair. By replacing the compromised key pair, the system maintains the integrity and security of VCs, preventing unauthorized access and fraudulent activities.

5.2 Security of IoT Devices

The adversary can compromise an IoT device $d_i \in D_j$, enabling them to launch various attacks within the IoT network n_j. d_i provided with a valid VC and w_i can establish trusted communications with other entities that place trust in i_j. Therefore, i_j is responsible for identifying and detecting malicious activities, ensuring that VCs associated with compromised devices are removed from a_j. Several approaches have been proposed to detect compromised devices and malicious activities [8, 32]. However, the detection of compromised devices lies beyond the scope of this paper. We assume that the issuer can employ different methods and tools available in the literature for these purposes. i_j updates W_j and a_j, distributing them to D_j. As a result, malicious $d_i \in D_j$ will not be provided with valid witnesses, thereby preventing the attacker from successfully bypassing the revocation mechanism using a compromised d_i or its stolen VC.

5.3 Security of Communication

In n_j, the traffic may not be encrypted, thus, further broadening the attack surface. We consider two possible attack scenarios and corresponding countermeasures occurring during the update phase:

Selective VC Forwarding. One attack scenario involves the attacker intentionally not forwarding the VC containing the updated version of the accumulator value a_j to outdated devices. As a result, outdated devices may continue to trust devices whose VCs are no longer valid. However, it is important to note that an outdated device can still be updated by an honest device during communication initiation. Additionally, to address the concern of the adversary providing an outdated version of a_j, devices only update the value of a_j if the associated timestamp is fresher than the previous one. By incorporating timestamp-based checks, devices can ensure that updates are only applied when newer information is available, mitigating the risk of relying on outdated VCs.

Furthermore, compromised devices instead of sharing the latest version of a_j with benign devices, which includes their revoked VC, may provide another outdated version that does not include their revoked VC but is still fresher than the one held by the benign device. As described previously, given the mismatch between the version of the accumulator value, if the benign device supports direct retrieval, it can detect deceptive attacks when collecting the updated witness directly. However, in cases where direct retrieval is not supported, the device will accept the witnesses corresponding to that version of a_j, establishing mutual trust. It is important to note that while this scenario poses a significant threat, which also affects other approaches like CRL that lack real-time revocation freshness, the risk posed is time-limited. This is evidenced by the observation that most devices are updated within the first hour, as highlighted in Subsection 6.4. Once devices receive the updated accumulator, they become immune to deception, as the revoked VC will no longer be included in the accumulator value.

VC Theft and Masquerading. The attacker can steal the VC comprising the updated w_i of d_i, attempting to masquerade as the legitimate device. However, since the adversary does not possess the key pair associated with d_i, they cannot prove the ownership of the VC and establish trusted communication. Furthermore, in case of transmissions that involve sensitive

information, data can be encrypted using the public key of the receiving device.

6 Evaluation

In this section, we present a detailed evaluation of our approach. We designed a series of experiments to thoroughly assess the effectiveness and applicability of EVOKE. In particular, we considered the following scenarios:

- **Commodity IoT Devices**: To gauge the practicality of implementing EVOKE in real-world scenarios, we conducted experiments on various off-the-shelf IoT devices.
- **Hybrid Network Settings**: Since not all commodity devices support running customized code directly, we performed a set of experiments on an IoT network that comprises both off-the-shelf devices and Raspberry Pis to explore more customizable and broader scenarios.
- **Large-scale Analysis**: As it would be impractical to evaluate our proposed approach on thousands of devices, we devised a set of experiments to simulate a large-scale IoT network and evaluate the scalability of EVOKE.

The results of these experiments, along with detailed explanations, are provided in the subsequent subsections. Through this comprehensive evaluation, we aim to demonstrate the effectiveness, feasibility, scalability, and versatility of EVOKE in various IoT deployment scenarios.

6.1 Implementation Setup

We developed a prototype of EVOKE to evaluate its runtime performance and test its core functionalities. We utilized JavaScript and WebAssembly (WASM) [4] for the implementation in commodity IoT devices. To ensure consistency across all experiments, the same functions were also employed in all scenarios. To enable cryptographic operations, we leveraged the `crypto-wasm-ts` library developed by docknetwork [13]. This library provides a comprehensive set of cryptographic primitives and algorithms implemented using WASM. It ensures efficient and performant execution of cryptographic tasks within EVOKE. For managing DIDs and VCs, we utilized the `identity-wasm/web` library from IOTA [24]. This library offers robust functionality for creating, managing, and verifying DIDs and VCs. Finally, to handle the DLT, we employed the IOTA Tangle [41], a DAG that is specifically designed to cater to the unique requirements of IoT applications [57, 58]. All the results have been averaged over 1K runs. To ensure transparency and reproducibility, we have made our prototype implementation, along with all the experiments, available on GitHub[1].

[1] https://github.com/evokevc/EVOKE

Table 2: Commodity IoT devices used in the experiments and their specifications.

Device	Type	Processor	Memory	Operating System
LG Smart TV	Smart TV	LG Quad Core Processor	1.5GB RAM	WebOS
Amazon Echo Show 5	Home Assistant	MediaTek MT8163	1 GB RAM	Linux
Apple iPhone 12	Smartphone	Apple A14 Bionic	4GB RAM	iOS 16
Oculus Quest 2	Virtual Reality Headset	Qualcomm Snapdragon XR2	6GB RAM	Oculus OS

Table 3: Performance evaluation of EVOKE operations on commodity IoT devices.

Operation	LG Smart TV	Amazon Echo Show	Apple iPhone 12	Oculus Quest 2
Verify valid VC	477.44 ms	499.70 ms	12.62 ms	48.69 ms
Verify revoked VC	476.89 ms	498.67 ms	12.58 ms	47.89 ms

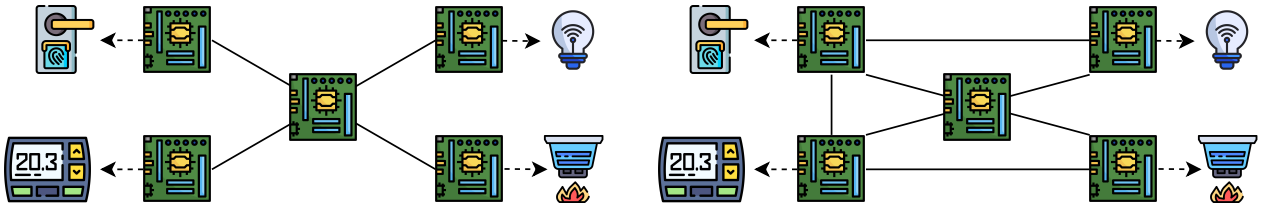
6.2 Commodity IoT Devices

In these experiments, we aim to evaluate the feasibility of using EVOKE on commodity IoT devices. For this, we set up an IoT network, which represents a modern smart home comprising different types of off-the-shelf IoT devices. Table 2 reports the specifications of the considered devices, which feature varying resources and operating systems. In our study, we considered devices capable of running EVOKE's code directly. Specifically, all the devices support JavaScript and WASM. To ensure compatibility with various devices, we realized different approaches. For instance, we developed a basic application using the LG WebOS developer framework [1, 48] for the LG Smart TV. For other devices, the code was provided through a Python web server.

Table 3 presents the results. In particular, we highlighted the performance of key operations across a range of commodity IoT devices. The results show that the iPhone 12 runs at about 13ms, while the Amazon Echo Show and LG Smart TV exhibit relatively slower performance. Nonetheless, all the devices verify VCs in under half a second. It is important to note that the size of the information is independent of the specific device used. The storage requirements for maintaining the accumulator value and the witness are approximately 1.5 KB. Notably, these values, along with the execution time of the main operations, remain consistent regardless of the number of VCs involved. This demonstrates the scalability and efficiency of EVOKE across different commodity devices in terms of resource utilization and execution time.

6.3 Hybrid Network Settings

In most cases, commodity devices do not provide direct access for running custom code. However, they often offer APIs that allow developers to control and interact with them. In this scenario, we considered a hybrid IoT network within a smart home, consisting of both off-the-shelf devices and Raspberry



(a) Star network topology

(b) Mesh network topology

Figure 4: Topologies used in our hybrid network experiments.

Pi, serving as controllers for the devices that only provide APIs. By leveraging the capabilities of Raspberry Pi, we were able to perform EVOKE's other operations. The Raspberry Pi devices act as intermediaries, enabling the execution of EVOKE functionalities on behalf of the devices. This approach allowed us to overcome the limitations imposed by the off-the-shelf devices' restricted programmability and harness the power of EVOKE's features. Moreover, this setup ensured the seamless integration of EVOKE operations across a range of devices, expanding the potential applications and scalability of the system. In modern smart homes, advanced integration and connectivity between devices enable enhanced safety and convenience features. For instance, consider a scenario where a smart lock is intelligently linked with a smoke detector. EVOKE allows establishing trust among the devices, enabling it to automatically unlock the door to facilitate the evacuation of occupants during a fire emergency.

In our experiments, we evaluated the performance of EVOKE in two popular network topologies for IoT protocols (i.e., Zigbee and Z-Wave) and scenarios [59]. Specifically, we considered star network topology and mesh network topology, whose deployments are depicted in Figure 4. These scenarios represent real-world use cases and provide valuable insights into the effectiveness and applicability of EVOKE in real IoT environments. The star topology (Figure 4a) consists of a central hub or controller connected to multiple peripheral devices. In a smart home scenario, it is usually represented by a home assistant (e.g., Amazon Alexa or Google Home). All communication flows through the central hub, which acts as a centralized point of control and coordination. This topology offers simplicity in management and facilitates efficient communication between devices and the central hub. However, it can pose limitations in terms of scalability and fault tolerance, as the entire network relies on the central component.

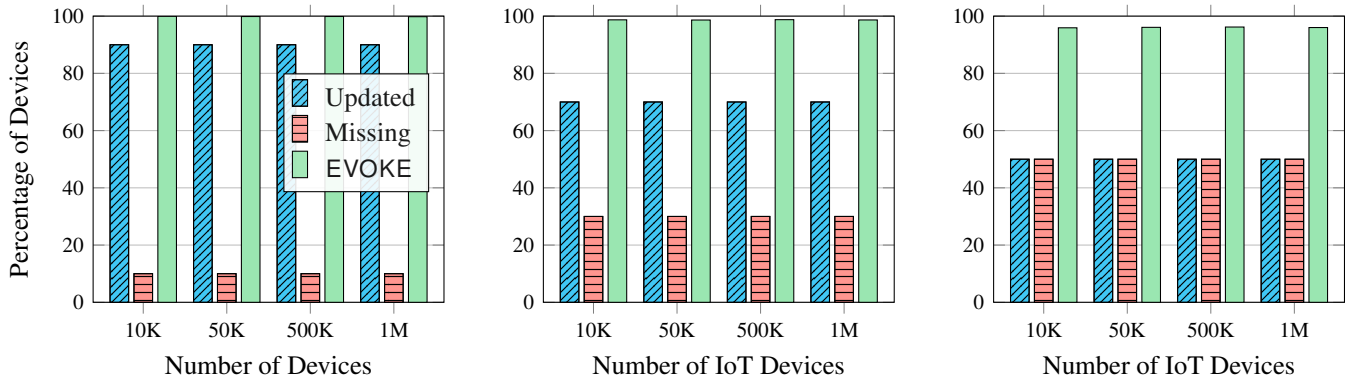
The mesh topology (Figure 4b) involves interconnecting multiple devices in a decentralized manner. Specifically, we considered a fully connected network, a particular case of a mesh network where each device can communicate directly with all the other devices. This topology provides robustness and redundancy, as there are multiple paths for data transmission. It can adapt well to dynamic environments and enables better fault tolerance. However, the complexity of managing and coordinating communication among numerous devices increases with the size of the network. By considering both the star and mesh topologies in our experiments, we aimed to assess the performance and suitability of EVOKE in different network configurations.

Table 4: Performance evaluation of EVOKE operations on commodity IoT devices with different network topologies.

Topology	Approach	Total Latency (Verify + Transfer)	E2E Latency
Star Network	EVOKE	1152.7 ms	948.3 ms
	Baseline	967.7 ms	705.5 ms
Mesh Network	EVOKE	545.2 ms	307.5 ms
	Baseline	97.4 ms	91.7 ms

To evaluate the performance of the two network topologies, we implemented them using a local cluster consisting of five Raspberry Pi4 provided with 8GB RAM. All Raspberry Pis were connected to the same Wi-Fi network, ensuring a controlled environment for our experiments. This setup allowed us to observe the network configurations and measure latency with precision. However, due to the inherent variations in clock accuracy among different devices, it was essential to synchronize their timestamps accurately to obtain reliable metrics in our experiments. To achieve this, we utilized the Network Time Protocol (NTP) [36], a widely adopted networking protocol specifically designed for clock synchronization in distributed networks.

In Table 4, we present the results for the two network topologies examined in our study. We focus on the key metrics: total latency and end-to-end (E2E) latency. The total latency encompasses the overall delay in establishing trust between two updated devices. On the other hand, E2E latency specifically captures the time taken for the verification data to traverse between devices. As expected, the star network exhibits latency more than double that of the mesh network. This discrepancy arises due to the higher number of devices involved in interactions within the star network. Notably, network overhead, including synchronization delay introduced by NTP, is the main factor that affects the overall latency. To evaluate the EVOKE's overhead, we compared it to a *baseline*, which represents latencies when sending a minimal amount of data across the same network configurations without perform-



(a) 10% devices missing updates (b) 30% devices missing updates (c) 50% devices missing updates

Figure 5: Percentage of updated IoT devices by EVOKE with varying the number of devices that miss updates with 5 interactions per device per hour.

ing any operations. It is worth noting that since the baseline only involves a minimal amount of data, it is expected to have better performance than any other revocation mechanism. EVOKE operations for verifying VCs only contribute a few milliseconds per device, posing minimal impact on total latency. Therefore, the increased latency between EVOKE and the baseline, in both configurations, is due to the transfer of revocation information. However, EVOKE's required size is much lower than existing approaches.

6.4 Large-scale Analysis

To conduct a scalability evaluation of EVOKE, an extensive number of devices would be required. However, due to practical limitations, we implemented a large-scale network simulation of IoT devices. The simulation was run on an 11th Gen Intel(R) Core(TM) i7-11370H @ 3.30GHz equipped with 4 cores and 16GB of RAM. The configurations we considered involved scaling the number of devices from 10,000 up to 1 million over a simulated week.

In addition to assessing scalability, our experiments also focused on evaluating the effectiveness of offline updates. To achieve this, we categorized devices as either updated or outdated, with varying percentages of devices missing updates. Additionally, an outdated device can be normal or constrained, the former supports direct retrieval, while constrained devices can only be updated indirectly. We adopted a revocation share of approximately 0.028% of VCs per day (almost 10% yearly), based on similar studies [27] that addressed PKI-certificate revocation. This estimation was derived from the impact of the Heartbleed bug on certificates [31]. When the issuer revokes a VC, the accumulator value and witnesses are updated and disseminated through the Tangle. However, due to various circumstances such as lack of connectivity or devices being in power-saving hibernation mode, a portion of IoT devices may not receive these updates. In our simulations, we considered different missing shares of 10%, 30%, and 50% in different experiment executions to understand the impact of outdated devices on the system. The number of interactions among devices depends on several factors including purpose, communication protocols, and the nature of the data being processed. For our evaluation, we consider the same network set-up of [27], where each device in a satellite network communicates with 5 random devices within an hour. Further experiments and considerations are presented in Appendix. During the interactions, devices exchange their accumulator values implementing the protocol described in Section 4. Therefore if a device has an outdated witness, the node with the older version requests an updated witness from the Tangle node or the interacting device if the capabilities are not sufficient.

Table 5: EVOKE's communication overhead to update offline devices with 5 interactions per device per hour.

Number of Devices	% Devices Missing Updates	Network Overhead (Accumulator + Witnesses)
10K	10	1.49 MB
	30	4.42 MB
	50	7.19 MB
50K	10	7.45 MB
	30	22.03 MB
	50	36.05 MB
500K	10	74.51 MB
	30	220.44 MB
	50	360.70 MB
1M	10	155.43 MB
	30	452.13 MB
	50	721.67 MB

In Figure 5, we highlight the percentage of updated IoT devices in one hour by EVOKE while varying the number of devices missing updates. The `Updated` bars indicate the percentage of devices that have received the novel accumulator value and witnesses, while the `Missing` bars denote the

percentage of devices missing such information. As clearly shown, the percentage of updated devices is not affected by the number of devices. The experimental results demonstrate that EVOKE can update the whole IoT network in the first hour when the number of devices missing updates is 10%. By increasing the number of disconnected devices to 30% and 50%, the percentage of updated devices in the network reached approximately 98% and 96%, respectively. Table 5 presents the network overhead, defined as the amount of data exchanged to updated devices that failed to receive updates. As expected, the overhead grows with both the number of devices and devices missing updates. This increase is due to the higher number of interactions that occur in the network.

Figure 6 illustrates, on a log scale, the three primary overheads encountered on the issuer side: the generation of the accumulator value, the production of witnesses for the included VCs, and the updating of the accumulator value. This latter comprises the removal of VCs and the generation of witnesses, which are obtained by the latest valid accumulator value. The update of the accumulator value was analyzed under various scenarios, considering different percentages of revoked VCs: 10%, 25%, and 50%. Evaluating EVOKE while revoking half of the issued VCs demonstrates its efficiency in handling mass revocations: the store requirements are not affected, while as the revoked VC percentage rises, the accumulator value time decreases due to fewer witnesses generated. This result is because updating the accumulator value involves the removal of VCs, which in turn reduces the number of witnesses to be generated. The graph highlights that the generation of witnesses is the most time-consuming operation. As demonstrated [5], the number of operations needed to generate w witnesses requires at least $\Omega(w)$ operations. However, even when generating 1 million witnesses, the process only requires 80K ms. It is important to note that issuers typically have more computational power, enabling a significant reduction in this overhead.

6.5 Discussion of Results

In this section, we present further considerations regarding the conducted experiments. As previously mentioned, to ensure consistency across all scenarios, including those involving commodity IoT devices, we implemented all experiments using the same JavaScript and WASM libraries. It is worth noting that employing diverse implementations and programming languages could potentially yield improved performance outcomes. These variations could lead to optimized execution and better resource utilization. Moreover, any revocation protocol is affected by the characteristics of the region in terms of latency and bandwidth. However, EVOKE's efficiency consists of minimizing storage and verification time on devices, hence, its performance is independent of the considered geographic area.

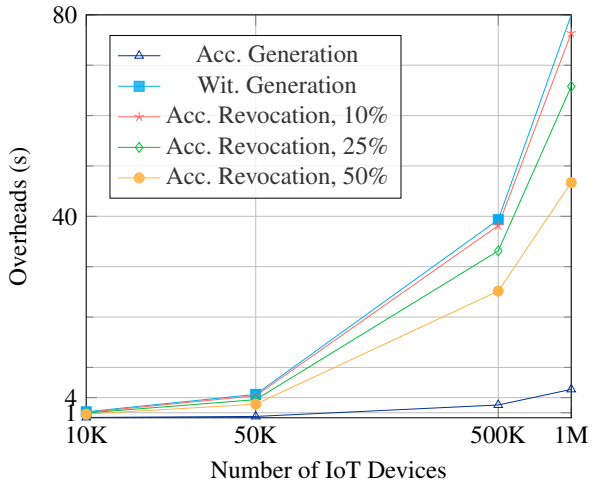


Figure 6: Large scale analysis: Issuer overhead for managing VCs.

Commodity IoT Devices. Vendors typically restrict end users from directly running code on their devices. To demonstrate the compatibility of EVOKE with commodity devices, we developed a JavaScript/WASM application. In this approach, a device sends a request to a server, which responds with an HTML page containing our JavaScript code. This allows the execution of the primary EVOKE operations on the device itself. It is important to note that we had to consider commodity devices with a browser connection to run EVOKE through JavaScript/WASM. Nevertheless, we demonstrated that vendors have the opportunity to integrate EVOKE into their products. The results reported in Tables 3 and 4 demonstrate that EVOKE already achieves remarkable performance when implemented in JavaScript. This underscores its potential for enhanced performance if vendors permit its native execution on devices. Additionally, the ECC-based accumulator used by EVOKE requires very limited storage and efficient verification times, making it particularly suitable for devices with constrained capabilities.

Hybrid Network Settings. The adoption of a hybrid network, which combines both off-the-shelf devices and single-board computers such as Raspberry Pis, provides a holistic approach to exploring the potential of our solution. This setup not only demonstrates the versatility of EVOKE but also broadens its scope, making it applicable in contexts where different types of devices coexist and require efficient and secure management of VCs. Through this set of experiments, we demonstrated the applicability of EVOKE even when the code is not executed directly on the target device but rather on the controlling board. This flexibility opens up new possibilities for deploying our solution in diverse environments. Moreover, the introduction of a controller in the network further expands the potential applications of EVOKE. By incorporating a controller into the system, we can extend the utilization

of VCs and, consequently, the use of EVOKE to encompass more resource-constrained devices. This capability becomes especially relevant for incorporating small sensors that lack the computational power to handle VCs or any revocation mechanisms directly.

Large-Scale Analysis. This set of experiments allowed us to demonstrate two important features of EVOKE that can be achieved through the accumulator: mass and offline revocation. The size of the accumulator value does not increase with the number of devices, thus, it can be employed for massively revoking VCs in any IoT network. The information stored by each device for the accumulator value and witness is always in the order of 1.5 KB. Including the accumulator value within a VC signed by the issuer and the reduced size of the accumulator value itself enable updated devices to update the ones that missed fresh revocation information. Also when 50% of devices are missing updates, after 1 hour, around 96% of the whole network has been successfully updated. As shown in Section 7, the amount of data related to revocation of EVOKE is minimal compared to other approaches. Figure 6 demonstrates that the main overhead is given by the generation and dissemination of witnesses. However, the issuer has the necessary resources to efficiently compute them. Notably, these overheads do not impact the EVOKE's performance in terms of storage and verification time on devices, which is the primary objective of our revocation scheme.

7 Comparison to Other Approaches

This section compares EVOKE to the state-of-the-art techniques in the revocation space and V'CER [27], which we consider the closest work in the literature. It is worth noting that these approaches are applied to PKI certificates, but they can be extended to VCs. An overview of the approaches discussed in this section is reported in Table 6. The storage column indicates the size required per device to store the whole revocation information of 1 million certificates or VCs, according to the focus of the work.

OCSP. In OCSP [42], the verifier sends a request to the issuer to check the validity of the presented VC. The traditional OCSP approach assumes the verifier has a fast and reliable connection to the issuer. However, this assumption may not hold in IoT networks due to the network overhead and the limited connectivity capabilities of certain devices. OCSP Stapling offers a more efficient alternative. It shifts the responsibility of querying the issuer from the verifier to the presenter of the VC. The presenter periodically queries the issuer and appends a time-stamped OCSP response, which is signed by the issuer. Thus, the verifier can obtain the necessary information about the VC's validity without additional communication. While OCSP Stapling provides a more efficient approach, it is worth noting that the stapled OCSP responses have a limited validity period and do not guarantee information freshness, which is a key advantage of this protocol. Furthermore, each OCSP request typically consumes around 4 KB of data [31], which exceeds the 1.5 KB required by EVOKE. Recently, TinyOCSP [22] managed to reduce the size by approximately 70%; however, meeting this requirement while maintaining a reliable connection remains challenging in many IoT networks.

Table 6: Comparison of EVOKE to other approaches

Work	Storage 1M Cert/VC	Mass Revocation	Offline Revocation
EVOKE	1.5 KB	✓	✓
V'CER [27]	3 KB	×	✓
Revoc. List 2020 [53]	125 KB	×	×
CRLite [45]	112.5 KB	×	×
Let's Revoke [30]	70 KB	×	×
TinyOCSP [22]	×	×	×

CRLs. When using CRL [10], devices need to store the whole list. However, in IoT networks comprising many devices and VCs, the amount of data to memorize can be extremely large. A CRL that contains only strictly necessary information for 1 million certificates would require at least a few MBs of storage [31].

Concerning VCs, the only revocation mechanism proposed by W3C is the draft namely Revocation List 2020 [53]. It associates each VC with a position in a bitstring managed by the issuer. If the binary value at a specific position is set to 1, the VC is considered revoked; otherwise, it is considered valid. With 1 million VCs, this would result in approximately 125 KB of storage. However, the use of a bitstring offers the advantage of high compressibility since a significant number of credentials often remain unrevoked, leading to long sections of repetitive bits. By employing compression techniques like ZLIB [12], the compressed size of the bitstring can be nearly halved, assuming an average compression ratio of 50% in case of mass revocation. Although a lower percentage of revoked VCs would yield a higher compression ratio, the memory storage would still exceed that of EVOKE. Indeed, EVOKE always requires that each device only stores 1.5 KB, even though the number of devices in the network may be larger. There are ongoing research efforts, such as CRLite [45] and Let's Revoke [30], aimed at reducing the storage and update overhead specifically for the Web's Public Key Infrastructure PKI. These solutions can reduce the size of the CRL to hundreds of kilobytes.

V'CER. In the literature, we find V'CER as the closest work to EVOKE. V'CER aims to tackle the challenge of efficient trust establishment on constrained networks using PKI certificates. V'CER adopts Sparse Merkle Trees (SMTs) [11], a type of MT that contains all possible hash values. Similar to our approach, V'CER concatenates valid certificates to create a hash root, which can be considered analogous to the final value of the accumulator in EVOKE. However, V'CER employs a small set of hashes called co-path as witnesses, which represents all sibling nodes on the path to the root and serves

as proof of inclusion. The size of this proof is $O(\log(n))$, where n is the number of leaves. Consequently, the storage required by devices may increase with the number of certificates. For instance, when dealing with 1 million certificates, the storage demand is approximately 3 KB. Although V'CER offers a valuable solution tailored for constrained devices, it demands twice the amount of data per device when compared to EVOKE. However, it offers the advantage of enabling devices to distributively repair their proofs in certain cases, without relying on CAs or delegates and supporting offline revocation. In contrast, EVOKE is the most efficient in terms of storage requirements since it demands around 1.5KB of storage and facilitates mutual assistance among devices to update.

Mass and Offline Revocation. Traditional approaches and their enhancements are unsuitable for IoT networks due to their storage and connection requirements, lacking adequate support for mass and offline revocation. By leveraging the features of the accumulator, EVOKE meets the requirements of mass revocation (Definition 1). On the contrary, related works do not meet them due to the usage of data structures whose size grows with the increase of the number of revoked VCs in the network, resulting in a longer verification time.

According to Definition 2, to support offline revocation, a revocation scheme must minimize the amount of data transmission to facilitate the sharing of revocation information. Other approaches employ bigger data structures that cause remarkable bandwidth overhead and make them unsuitable for constrained networks. For example, CRLs demand roughly two orders of magnitude greater than EVOKE for data transmission. Additionally, in the event of missed updates, these methods always rely on devices directly contacting the issuer or their delegates. In contrast, the very limited storage requirements of EVOKE as well as the use of VCs to share revocation information empower updated devices to bring outdated ones up to date.

8 Related Work

Despite the growing interest in VCs, there is a limited research that focuses on developing novel and efficient approaches for their revocation. In this section, we reviewed the existing literature on the revocation of VCs and related works for certificates/credentials, which can be eventually extended to VC, with a particular emphasis on IoT networks.

Verifiable Credential Revocation. Most studies lack a specific emphasis on constrained environments and use traditional revocation mechanisms [9]. For instance, Abraham et al. [2] proposed to revoke VCs by including them in a revocation list. To allow offline verification, nodes (i.e., issuers and users) can generate attestations for valid VCs. These attestations enable verifiers to determine the freshness of the presented credential, ensuring its validity. Recently, Fotiu et al. [18] introduced a lightweight revocation mechanism for VCs tailored for IoT environments, inspired by the W3C draft [53]. The proposed approach involves maintaining a revocation list, where each VC is assigned a position. The position information is embedded within each VC. If the corresponding position is set to 1, the VC is revoked. It is worth noting that this mechanism is most suitable for scenarios with a restricted number of VCs. A similar approach is also presented in [19], wherein in a group of IoT devices, each member is allowed to participate through a VC. The revocation list is included in a TXT DNS record a retrieved is identified through DNS resolution. This solution specifically refers to group member revocation and may not be suitable for use cases where devices are not clustered in groups.

Accumulator-based Approaches. Cryptographic accumulators have long been recognized as an efficient solution for revoking anonymous credentials and PKI certificates [6, 21]. However, most of the existing literature has primarily focused on user-centric revocation rather than efficient revocation in constrained networks. For example, in the protocol recently proposed by Parameswarath et al. [39], the accumulator is used to revoke user policy. The unique characteristics of accumulators, such as their constant memory size and verification time, harness them to design novel revocation schemes specifically tailored for IoT networks [7, 56]. In this direction, the most relevant related work is V'CER [27], an efficient certificate validation in constrained networks. V'CER adopts SMTs, a form of a hash-based accumulator, for performing revocation. The proposed approach achieves remarkable storage efficiency, requiring less than 3KB per node to store 1 million certificates, while also supporting offline updates.

DLT-based Approaches. Numerous studies [28, 44, 59] have highlighted the potential of DLTs in revolutionizing PKI-based authentication and authorization schemes. One particular area where blockchain technology has shown promise is in the management of certificates, with initial applications focusing on storing certificate updates and revocation information [16]. However, implementing a blockchain-based revocation system for IoT devices presents several challenges, including long latency, high energy consumption, and low transaction throughput. To address these challenges, innovative approaches [49, 57] leveraging DAG-based ledgers have emerged as a viable alternative for IoT networks. DAG-based ledgers offer a lightweight solution that maintains the same level of security as traditional blockchains while addressing the specific requirements of IoT environments.

Differences from Existing Works. While several existing approaches can be adapted for revoking VCs, they are often not suitable for constrained environments of IoT networks. In this context, EVOKE stands out as the first revocation scheme specifically designed for VCs in IoT networks. EVOKE offers offline revocation capabilities similar to V'CER, but with even lower storage overhead. Each device only requires less

than 1.5KB to store the accumulator value and the witness. Furthermore, EVOKE introduces the advantage of supporting mass revocation. Unlike traditional schemes, the size of the accumulator value and witness remains constant regardless of the number of included VCs.

9 Conclusion

VCs offer a promising solution to enhance trust among IoT devices. However, ensuring the efficient revocation of VCs in IoT networks remains a critical challenge that is still in its early stages of research. Our work lays the foundation for future research and developments in this field.

In this paper, we presented EVOKE, an innovative solution that addresses the efficient revocation of VCs in IoT networks. EVOKE harnesses the power of ECC-based accumulators to efficiently manage VCs with minimal computational and storage overhead. It also provides essential features like mass and offline revocation, making it a scalable and efficient revocation scheme for IoT networks. To comprehensively validate its effectiveness, we conducted extensive experiments comprising commodity IoT devices, hybrid network settings, and large-scale analysis. The results showcased the efficiency and practicality of EVOKE in real-world scenarios. Each device only demands 1.5 KB of storage for verification information. The experiments conducted in the hybrid network settings showed an overhead in the order of milliseconds for performing key operations. Finally, the large-scale analysis demonstrated its feature to update devices that miss revocation information.

Acknowledgments

We thank the anonymous reviewers and our shepherd for their helpful feedback and dedication. This work was partially supported by the project SERICS (PE00000014) under the MUR National Recovery and Resilience Plan program funded by the European Union - NextGenerationEU, the US National Science Foundation (Award: 2219920), and Microsoft. The views expressed are those of the authors only, not of the funding agencies.

References

[1] LG WebOS TV Developer. Accessed: 2023-10-06.

[2] ABRAHAM, A., MORE, S., RABENSTEINER, C., AND HÖRANDNER, F. Revocable and Offline-Verifiable Self-Sovereign Identities. In *2020 IEEE 19th International Conference on Trust, Security and Privacy in Computing and Communications (TrustCom)* (2020), pp. 1020–1027.

[3] BENALOH, J., AND DE MARE, M. One-way accumulators: A decentralized alternative to digital signatures. In *Advances in Cryptology — EUROCRYPT '93* (Berlin, Heidelberg, 1994), T. Helleseth, Ed., Springer Berlin Heidelberg, pp. 274–285.

[4] BHANSALI, S., ARIS, A., ACAR, A., OZ, H., AND ULUAGAC, A. S. A First Look at Code Obfuscation for WebAssembly. In *Proceedings of the 15th ACM Conference on Security and Privacy in Wireless and Mobile Networks* (New York, NY, USA, 2022), WiSec '22, Association for Computing Machinery, p. 140–145.

[5] CAMACHO, P., AND HEVIA, A. On the impossibility of batch update for cryptographic accumulators. In *Progress in Cryptology–LATINCRYPT 2010: First International Conference on Cryptology and Information Security in Latin America, Puebla, Mexico, August 8-11, 2010, proceedings 1* (2010), Springer, pp. 178–188.

[6] CAMENISCH, J., KOHLWEISS, M., AND SORIENTE, C. An Accumulator Based on Bilinear Maps and Efficient Revocation for Anonymous Credentials. In *Public Key Cryptography – PKC 2009* (Berlin, Heidelberg, 2009), S. Jarecki and G. Tsudik, Eds., Springer Berlin Heidelberg, pp. 481–500.

[7] CEBE, M., AND AKKAYA, K. Communication-efficient certificate revocation management for Advanced Metering Infrastructure and IoT Integration. *Future Generation Computer Systems 115* (2021), 267–278.

[8] CHAABOUNI, N., MOSBAH, M., ZEMMARI, A., SAUVIGNAC, C., AND FARUKI, P. Network Intrusion Detection for IoT Security Based on Learning Techniques. *IEEE Communications Surveys & Tutorials 21*, 3 (2019), 2671–2701.

[9] CHADWICK, D. W., LABORDE, R., OGLAZA, A., VENANT, R., WAZAN, S., AND NIJJAR, M. Improved Identity Management with Verifiable Credentials and FIDO. *IEEE Communications Standards Magazine 3*, 4 (2019), 14–20.

[10] COOPER, D., SANTESSON, S., FARRELL, S., BOEYEN, S., HOUSLEY, R., AND POLK, W. Internet x. 509 public key infrastructure certificate and certificate revocation list (crl) profile. Tech. rep., 2008.

[11] DAHLBERG, R., PULLS, T., AND PEETERS, R. Efficient Sparse Merkle Trees. In *Secure IT Systems* (Cham, 2016), B. B. Brumley and J. Röning, Eds., Springer International Publishing, pp. 199–215.

[12] DEUTSCH, P., AND GAILLY, J.-L. Zlib compressed data format specification version 3.3. Tech. rep., 1996.

[13] DOCKNETWORK. crypto-wasm-ts.

[14] DOLEV, D., AND YAO, A. On the security of public key protocols. *IEEE Transactions on Information Theory 29*, 2 (1983), 198–208.

[15] EISENSTADT, M., RAMACHANDRAN, M., CHOWDHURY, N., THIRD, A., AND DOMINGUE, J. Covid-19 antibody test/vaccination certification: There's an app for that. *IEEE Open Journal of Engineering in Medicine and Biology 1* (2020), 148–155.

[16] ELLOH ADJA, Y. C., HAMMI, B., SERHROUCHNI, A., AND ZEADALLY, S. A blockchain-based certificate revocation management and status verification system. *Computers & Security 104* (2021), 102209.

[17] FARAHANI, B., FIROUZI, F., AND LUECKING, M. The convergence of IoT and distributed ledger technologies (DLT): Opportunities, challenges, and solutions. *Journal of Network and Computer Applications 177* (2021), 102936.

[18] FOTIOU, N., SIRIS, V. A., POLYZOS, G. C., KORTESNIEMI, Y., AND LAGUTIN, D. Capabilities-based access control for IoT devices using Verifiable Credentials. In *2022 IEEE Security and Privacy Workshops (SPW)* (2022), pp. 222–228.

[19] FOTIOU, N., SIRIS, V. A., XYLOMENOS, G., AND POLYZOS, G. C. IoT Group Membership Management Using Decentralized Identifiers and Verifiable Credentials, journal = Future Internet.

[20] GHOSH, B. C., PATRANABIS, S., VINAYAGAMURTHY, D., RAMAKRISHNA, V., NARAYANAM, K., AND CHAKRABORTY, S. Private Certifier Intersection. *Cryptology ePrint Archive* (2022).

[21] GOODRICH, M. T., TAMASSIA, R., AND HASIĆ, J. An efficient dynamic and distributed cryptographic accumulator. In *Information Security: 5th International Conference, ISC 2002 Sao Paulo, Brazil, September 30–October 2, 2002 Proceedings 5* (2002), Springer, pp. 372–388.

[22] HÖGLUND, J., FURUHED, M., AND RAZA, S. Lightweight certificate revocation for low-power IoT with end-to-end security. *Journal of Information Security and Applications 73* (2023), 103424.

[23] HÖGLUND, J., LINDEMER, S., FURUHED, M., AND RAZA, S. PKI4IoT: Towards public key infrastructure for the Internet of Things. *Computers & Security 89* (2020), 101658.

[24] IOTA FUNDATION. IOTA Identity Framework.

[25] KANTARA INITAITIVE. Identities of Things Discussion Group.

[26] KIM, S., ZHANG, A., LIAO, R., ZHENG, W., HU, Z., AND SUN, Z. Sampling blockchain-enabled smart city applications among South Korea, the United States and China. *Journal of Smart Cities and Society 1*, 1 (2022), 53–70.

[27] KOISSER, D., JAUERNIG, P., TSUDIK, G., AND SADEGHI, A.-R. V'CER: Efficient Certificate Validation in Constrained Networks. In *31st USENIX Security Symposium (USENIX Security 22)* (Boston, MA, Aug. 2022), USENIX Association, pp. 4491–4508.

[28] KUBILAY, M. Y., KIRAZ, M. S., AND MANTAR, H. A. CertLedger: A new PKI model with Certificate Transparency based on blockchain. *Computers & Security 85* (2019), 333–352.

[29] KUMAR, A., LAFOURCADE, P., AND LAURADOUX, C. Performances of cryptographic accumulators. In *39th Annual IEEE Conference on Local Computer Networks* (2014), pp. 366–369.

[30] LARISCH, J., CHOFFNES, D., LEVIN, D., MAGGS, B. M., MISLOVE, A., AND WILSON, C. CRLite: A Scalable System for Pushing All TLS Revocations to All Browsers. In *2017 IEEE Symposium on Security and Privacy (SP)* (2017), pp. 539–556.

[31] LIU, Y., TOME, W., ZHANG, L., CHOFFNES, D., LEVIN, D., MAGGS, B., MISLOVE, A., SCHULMAN, A., AND WILSON, C. An End-to-End Measurement of Certificate Revocation in the Web's PKI. In *Proceedings of the 2015 Internet Measurement Conference* (New York, NY, USA, 2015), IMC '15, Association for Computing Machinery, p. 183–196.

[32] LIU, Y., WANG, J., LI, J., NIU, S., AND SONG, H. Machine Learning for the Detection and Identification of Internet of Things Devices: A Survey. *IEEE Internet of Things Journal 9*, 1 (2022), 298–320.

[33] MARAM, D., MALVAI, H., ZHANG, F., JEAN-LOUIS, N., FROLOV, A., KELL, T., LOBBAN, T., MOY, C., JUELS, A., AND MILLER, A. CanDID: Can-Do Decentralized Identity with Legacy Compatibility, Sybil-Resistance, and Accountability. In *2021 IEEE Symposium on Security and Privacy (SP)* (2021), pp. 1348–1366.

[34] MAZZOCCA, C., ACAR, A., ULUAGAC, S., MONTANARI, R., BELLAVISTA, P., AND CONTI, M. A Survey on Decentralized Identifiers and Verifiable Credentials. *arXiv preprint arXiv:2402.02455* (2024).

[35] MAZZOCCA, C., ROMANDINI, N., MENDULA, M., MONTANARI, R., AND BELLAVISTA, P. TruFLaaS: Trustworthy Federated Learning as a Service. *IEEE Internet of Things Journal* (2023), 1–1.

[36] MILLS, D. Internet time synchronization: the network time protocol. *IEEE Transactions on Communications 39*, 10 (1991), 1482–1493.

[37] NEMEC, M., SYS, M., SVENDA, P., KLINEC, D., AND MATYAS, V. The Return of Coppersmith's Attack: Practical Factorization of Widely Used RSA Moduli. In *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security* (New York, NY, USA, 2017), CCS '17, Association for Computing Machinery, p. 1631–1648.

[38] NÓBREGA GONÇALVES, S. M., TOMASI, A., BISEGNA, A., PELLIZZARI, G., AND RANISE, S. Verifiable Contracting: A Use Case for Onboarding and Contract Offering in Financial Services with eIDAS and Verifiable Credentials. In *Computer Security: ESORICS 2020 International Workshops, DETIPS, DeSECSys, MPS, and SPOSE, Guildford, UK, September 17–18, 2020, Revised Selected Papers 25* (2020), Springer, pp. 133–144.

[39] PARAMESWARATH, R. P., GOPE, P., AND SIKDAR, B. A Privacy-Preserving Authenticated Key Exchange Protocol for V2G Communications Using SSI. *IEEE Transactions on Vehicular Technology* (2023), 1–16.

[40] PODGORELEC, B., ALBER, L., AND ZEFFERER, T. What is a (Digital) Identity Wallet? A Systematic Literature Review. In *2022 IEEE 46th Annual Computers, Software, and Applications Conference (COMPSAC)* (2022), pp. 809–818.

[41] POPOV, S. The tangle. *White paper 1*, 3 (2018), 30.

[42] SANTESSON, S., MYERS, M., ANKNEY, R., MALPANI, A., GALPERIN, S., AND ADAMS, C. X. 509 internet public key infrastructure online certificate status protocol-ocsp. Tech. rep., 2013.

[43] SHAFIQUE, K., KHAWAJA, B. A., SABIR, F., QAZI, S., AND MUSTAQIM, M. Internet of Things (IoT) for Next-Generation Smart Systems: A Review of Current Challenges, Future Trends and Prospects for Emerging 5G-IoT Scenarios. *IEEE Access 8* (2020), 23022–23040.

[44] SINGLA, A., AND BERTINO, E. Blockchain-Based PKI Solutions for IoT. In *2018 IEEE 4th International Conference on Collaboration and Internet Computing (CIC)* (2018), pp. 9–15.

[45] SMITH, T., DICKINSON, L., AND SEAMONS, K. Let's Revoke: Scalable Global Certificate Revocation. *Network and Distributed Systems Security (NDSS) Symposium 2020*.

[46] STATISTA. Number of Internet of Things (IoT) connected devices worldwide from 2019 to 2023, with forecasts from 2022 to 2030. Accessed: 2023-10-12.

[47] SZYDLO, M. Merkle tree traversal in log space and time. In *Eurocrypt* (2004), vol. 3027, Springer, pp. 541–554.

[48] TEKINER, E., ACAR, A., AND ULUAGAC, A. S. A lightweight IoT cryptojacking detection mechanism in heterogeneous smart home networks. In *Proc. of the ISOC Network and Distributed System Security Symposium (NDSS)* (2022).

[49] TESEI, A., DI MAURO, L., FALCITELLI, M., NOTO, S., AND PAGANO, P. IOTA-VPKI: A DLT-Based and Resource Efficient Vehicular Public Key Infrastructure. In *2018 IEEE 88th Vehicular Technology Conference (VTC-Fall)* (2018), pp. 1–6.

[50] TREMEL, E. Real-world performance of cryptographic accumulators. *Undergraduate Honors Thesis, Brown University 78* (2013).

[51] U.S. DEPARTMENT OF HOMELAND SECURITY. News Release: DHS ST Seeks Solutions for Privacy Preserving Digital Credential Wallets Verifiers .

[52] VITTO, G., AND BIRYUKOV, A. Dynamic universal accumulator with batch update over bilinear groups. In *Topics in Cryptology–CT-RSA 2022: Cryptographers' Track at the RSA Conference 2022, Virtual Event, March 1–2, 2022, Proceedings* (2022), Springer, pp. 395–426.

[53] W3 RECOMMENDATION. Revocation List 2020.

[54] W3 RECOMMENDATION. Decentralized Identifiers (DIDs) v1.0.

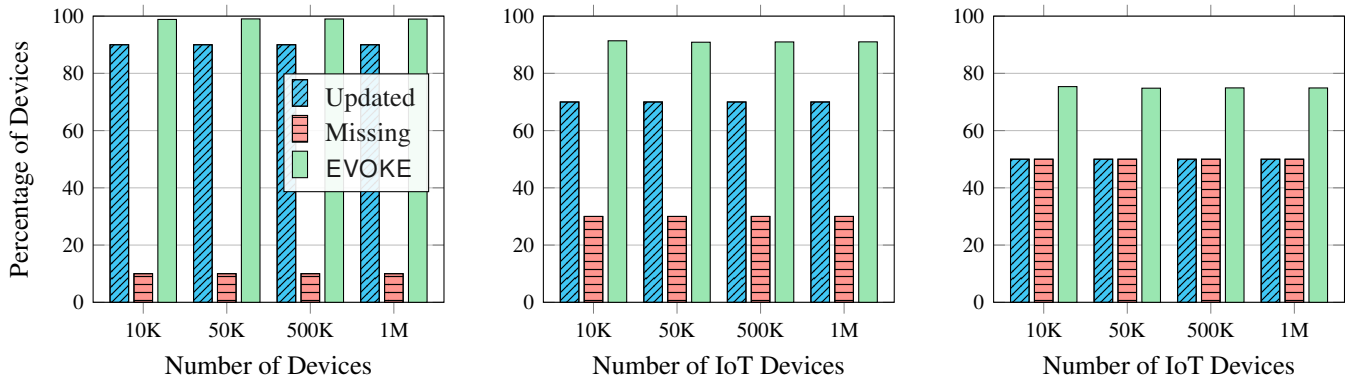
[55] W3 RECOMMENDATION. Verifiable Credentials Data Model v1.1.

[56] WANG, L., TIAN, Y., AND ZHANG, D. Toward Cross-Domain Dynamic Accumulator Authentication Based on Blockchain in Internet of Things. *IEEE Transactions on Industrial Informatics 18*, 4 (2022), 2858–2867.

[57] WANG, S., LI, H., CHEN, J., WANG, J., AND DENG, Y. DAG blockchain-based lightweight authentication and authorization scheme for IoT devices. *Journal of Information Security and Applications 66* (2022), 103134.

[58] WANG, T., WANG, Q., SHEN, Z., JIA, Z., AND SHAO, Z. Understanding Characteristics and System Implications of DAG-Based Blockchain in IoT Environments. *IEEE Internet of Things Journal 9*, 16 (2022), 14478–14489.

[59] ZHU, Q., LOKE, S. W., TRUJILLO-RASUA, R., JIANG, F., AND XIANG, Y. Applications of Distributed Ledger Technologies to the Internet of Things: A Survey. *ACM Comput. Surv. 52*, 6 (nov 2019).



(a) 10% devices missing updates (b) 30% devices missing updates (c) 50% devices missing updates

Figure 7: Percentage of updated IoT devices by EVOKE with varying the number of devices that miss updates with 1 interaction per device per hour.

Table 7: EVOKE's communication overhead to update offline devices with 1 interaction per device per hour.

Number of Devices	% Devices Missing Updates	Network Overhead (Accumulator + Witnesses)
10K	10	1.26 MB
	30	3.05 MB
	50	3.75 MB
50K	10	6.77 MB
	30	15.66 MB
	50	18.61 MB
500K	10	67.53 MB
	30	157.36 MB
	50	186.91 MB
1M	10	134.58 MB
	30	315.16 MB
	50	375.76 MB

Appendix

This section provides additional experiments and results for the large-scale analysis presented in Subsection 6.4.

A.1 Additional Experiments

To show how EVOKE performs in scenarios with lower interaction rates, we conducted additional experiments by setting the number of interactions per device per hour to 1. It is worth noting that EVOKE would perform even better in scenarios with higher interaction rates. Therefore, we did not consider configurations that foresee more than 5 interactions per hour. For consistency with the experiments reported in Subsection 6.4, we maintained the revocation share and the percentage of devices missing updates unchanged. Figure 7 depicts the percentage of updated IoT devices in one hour by EVOKE while varying the number of devices missing updates when the number of interactions per hour is set to 1. Similar to previous observations outlined in Subsection 6.4, the percentage of updated devices does not depend on the number of devices.

The experimental findings demonstrate that even in scenarios with lower interaction rates, EVOKE exhibits remarkable performance in updating the entire IoT network in the first hour when 10% of devices miss updates. However, as the number of disconnected devices increases to 30% and 50%, the percentage of updated devices decreases to approximately 91% and 75%, respectively. This demonstrates the robustness of EVOKE in updating the network despite a decrease in interaction frequency. Further insights into EVOKE's performance are presented in Table 7, detailing its communication overhead for updating offline devices in the examined configuration. As expected, the network overhead decreases with the number of interactions per device per hour.