AccA: A Decentralized and Accumulator-Based Authentication and Authorization Architecture for Autonomous IoT in Connected Infrastructures

Hannes Salin[a]
Swedish Transport Administration, Department of Information and Communication Technology, Borlänge, Sweden

Keywords: Trust Model, Trust Architecture, Cryptographic Accumulators, C-ITS, Authentication, Authorization.

Abstract: In the realm of Intelligent Transport Systems and connected infrastructures, the use of IoT devices offers improved safety and efficient traffic management. However, the emergence of trends such as Social IoT, particularly in ad-hoc networking, poses a significant challenge for cybersecurity and trust between nodes. To address this, we propose an efficient trust model architecture designed specifically for dynamic, ad-hoc environments where IoT interactions are frequent. Our model focuses on decentralized authorization, where trust is established on the object level, rather than relying on centralization. Our proposed architecture is backed by security proofs and a proof-of-concept implementation using nested cryptographic accumulators, which shows the effectiveness and feasibility of the proposed trust mechanism.

1 INTRODUCTION

Connected road- and railway infrastructures, such as Cooperative Intelligent Transport Systems (C-ITS), are complex environments with moving and stationary nodes. The main objective of C-ITS in the road vehicle industry is to facilitate communication between vehicles and the surrounding infrastructure, thus enhancing road safety and optimizing traffic flow (Zeddini et al., 2022). Furthermore, the integration of drones into C-ITS architectures has been explored as it could potentially broaden the range of potential applications (Valle et al., 2021). Internet of Things (IoT) and industrial IoT (IIoT) are also two major technologies within these types of environments. While the advent of new technology and connected infrastructures creates many opportunities for novel applications, there is a pressing need for secure architecture and trust models within these systems. This is because the large number of connections and the formation of relationships between nodes require robust authentication and authorization processes (Shahab et al., 2022; Galego and Pascoal, 2022). Numerous access control models (ACM) exist, and many of them are separated into different layers, e.g. cloud-, object-, and virtual layers (Gupta et al., 2022). Moreover, from an architectural perspective, the networking layers must also be considered for the access control, e.g. application- or transport layers in the TCP/IP stack. An ACM specifies how a user or object gains access to specific functions or capabilities in other objects. The Discretionary Access Control (DAC) model is an identity-based model where a user or object has complete control over its own resources, and control the permissions given to other users or objects. Usually this model is implemented as an access matrix, authorization table or access control list (Ravidas et al., 2019). Another model is the Role-based Access Control (RBAC) model which instead of identities grants permissions to predefined roles of users or objects. Several other type of ACM can be used as well, e.g. Attribute-based Access Control (ABAC) which restricts access via attributes, also expressed as policies (Ravidas et al., 2019).

1.1 Problem Statement

Trust management in IoT environments remains a challenge (Chahal et al., 2020; Khan et al., 2020; Sharma et al., 2020; Hammi et al., 2022). Our goal is to investigate how short-lived networks of collaborative interactions between IoT devices, where each device dictates its own authorization mechanism, can be provided using efficient and secure means.

[a] https://orcid.org/0000-0001-6327-3565

Salin, H.
AccA: A Decentralized and Accumulator-Based Authentication and Authorization Architecture for Autonomous IoT in Connected Infrastructures.
DOI: 10.5220/0011959200003482
In *Proceedings of the 8th International Conference on Internet of Things, Big Data and Security (IoTBDS 2023)*, pages 170-177
ISBN: 978-989-758-643-9; ISSN: 2184-4976

1.2 Related Work

The usage of cryptographic accumulators have previously been investigated in the domain of ad-hoc networking, particularly in the domain of C-ITS, however regarded in the context of authentication (Zuo et al., 2021; Salin et al., 2021), secret key storage (Salin and Fokin, 2022), vehicle pseudonymization (Förster et al., 2014) and vehicle certificate revocation (Heng et al., 2022). Although efficient solutions for authentication are proposed, no combined authentication and authorization mechanism was found, using a secure accumulator-based architecture. Lauinger et al. proposed an authorization scheme for Internet of Vehicles built on accumulators and zero-knowledge proofs, however, the architecture needs a managing root authority for the accumulators and not self-contained in each object, i.e. the vehicle (Lauinger et al., 2021). Finally, in the comprehensive study by Khan et al. on IoT-specific authorization schemes, no accumulator-based solutions were to be found either (Khan et al., 2022).

1.3 Contribution

Our contribution consists of a novel secure authentication and authorization architecture for ad-hoc and short-lived IoT environments. We also provide a security analysis and a proof of concept implementation with a performance analysis.

1.4 Structure of the Paper

In Sec. 2 we provide necessary preliminaries including systems settings and the threat model. In Sec. 3 we introduce the access control architecture with the proposed protocols. Sec. 4 provides a correctness and security analysis of the protocols and Sec. 5 elaborates on the proof of concept implementation. We conclude our paper in Sec. 6.

2 PRELIMINARIES

We refer to $\mathcal{H}$ as a secure hash function $\mathcal{H}: \{1,0\}^* \rightarrow \{1,0\}^n$ for some fixed n, resistant to preimage attacks. The output of $\mathcal{H}$, i.e. the hash digest $\mathcal{H}(m)$ is an element of a secure group $\mathbb{G}$. We denote the generator of $\mathbb{G}$ as g.

Definition 1 (Strong RSA Assumption)**.** *Let k be a security parameter. Given a k-bit RSA modulus N and a random element $x \in \mathbb{Z}_N^*$, there exists no probabilistic polynomial-time algorithm that outputs $y > 1$, and α such that $\alpha^y = x \bmod N$, except with negligible probability.*

The accumulator we use in our architecture is based the dynamic RSA-based scheme, proposed by Benaloh and De Mare (Benaloh and de Mare, 1994). However, our proposed architecture does not rely on a specific accumulator per se, but instead of detailing the architecture with an accumulator as a black box, we illustrate the overall architecture using the RSA-based scheme.

Definition 2. *Let p and q be strong primes. Let $a \in \mathbb{Z}$ be relatively prime to $pq = N$, secure under Def. 1. To accumulate element x_i the following computation is done: $\mathcal{Z} \leftarrow a^{x_i} \bmod N$. A witness extraction for x_i, i.e. w_i, we compute $w_i = a^{\prod_{j=1:j\neq i}^{n} x_j}$, over $\mathcal{Z}$ with n elements. To verify that $x_i \in \mathcal{Z}$ we compute $w_i^{x_i} \stackrel{?}{=} \mathcal{Z}$. To delete x_i we compute x_i^{-1}, then run $\mathcal{Z}' \leftarrow \mathcal{Z}^{x_i^{-1}}$.*

Thus, the scheme has three major procedures we consider in our architecture:

- $\mathsf{Acc}(x)$: accumulates element x into $\mathcal{Z}$ by exponentiation, as described in Def. 2. Also, this procedure returns the corresponding witness w when x is accumulated.
- $\mathsf{Del}(x)$: deletes element x_i from $\mathcal{Z}$ by accumulating the inverse to x_i, as described in Def. 2.
- $\mathsf{Ver}(w, x, \mathcal{Z})$: verifies if x_i is accumulated into $\mathcal{Z}$, returns 1 if accepted, 0 otherwise.

The accumulator we use in our proposal is proven secure under the strong RSA-assumption in Def. 1.

Boneh-Lynn-Shacham (BLS) short signatures (Boneh et al., 2001) are based on pairings and consists of a set of procedures and a key-pair $(\mathtt{sk}, \mathtt{pk})$ of private and public keys respectively (of the signer). The signature σ is computed as follows:

$$\mathcal{H}_g(m)^{\mathtt{sk}} = \sigma \tag{1}$$

where g is the chosen generator in $\mathbb{G}$. The signature is verified by checking that the equation

$$\hat{e}(\sigma, g) \stackrel{?}{=} \hat{e}(\mathcal{H}(m), \mathtt{pk}) \tag{2}$$

holds.

2.1 System Setting

We describe a typical C-ITS setting where IoT nodes and vehicles are referred to as *objects* - denoted $\mathcal{OBJ}$ - and humans in the system as *users*. A user always has an associated device, i.e. object, as means of communication with other users or objects. We denote an ad-hoc network with multiple connections to different $\mathcal{OBJ}$'s as NET. We highlight that a certain NET is a bounded logical network for a designated area, e.g. a smart home network, a clustered drone network or a VANET. Each NET have at least one authorized registration object that allows other objects to register to the NET. However, in contrast to isolated public key infrastructure (PKI) settings, object registration is the only function for the registration object. All subsequent authentications and authorizations are made by the participating objects themselves since the objects are the entities that provide the accessible capabilities within them. We assume standard communication between objects to be secured via encryption, except particular messages that according to standards are chosen un-encrypted (European Telecommunications Standards Institute, 2014). In a NET, different actors are differentiated, e.g. authorities (police, government etc.), private third parties (companies and business) and other objects. These actors are referred to as *types*. A certain object may have a standard trust setup with a limited number of types that can be handled. However, due to the ad-hoc environment in the NET, the authorization matrix the object possesses can be updated. The functions an object may give access to can be to share certain (vehicle, traffic, geographical or sensory) data, turning on/off different functions or allowing proxy connections for extended communication.

2.2 Threat Model

We consider a threat model, using adversaries that either observe the ad-hoc network of connections NET from the outside, or is part of the same network as a participant, hence adversary $\mathcal{A}$ is an object within the given system setting, as described in Sec. 2.1. $\mathcal{A}$ have the ability to capture and record data transfer between nodes in any given NET. $\mathcal{A}$ does not have the computational ability to steal or tamper memory data from other objects, nor can it access any secret keys. All other participants are honest, non-tampered objects $\mathcal{OBJ}_1, ..., \mathcal{OBJ}_k$. We define two security experiments for $\mathcal{A}$ in the given NET architecture:

- Exp_{A_1}: This experiment, denoted type A_1, is when the adversary tries to gain access to function f_i in object $\mathcal{OBJ}_k$ which it has not access to according to an authorization mechanism. However, $\mathcal{A}$ is part of the NET and otherwise trusted to access other functions of the target object.
- Exp_{A_2}: This experiment, denoted type A_2, is when the adversary is not part of the NET but tries to get access to function f_i in object $\mathcal{OBJ}_k$, hence there is no storage of any keys nor data of $\mathcal{A}$ in $\mathcal{OBJ}_k$.

We note that the index of the object is k, but w.l.o.g it can be any participant, thus we fix the targeted object to have index k for simplicity. Exp_{A_1} represents an attack which is very plausible due to dishonest users or objects that are compromised; by accessing the internal software, hardware tampering or lack of user security awareness (Gangolli et al., 2022; Polychronou et al., 2021; Sadhu et al., 2022).

3 THE ACCESS CONTROL ARCHITECTURE

Our architecture provides authorization without the need for a central party, except initial registration for a certain area that this party handles. Instead, as trusted parties in the environment after registration, all objects can handle the authorization separately based on the individual needs. The ACM that is embedded in our architecture is thus a DAC and RBAC hybrid where access is granted by the object itself, and based on roles (types). We denote the model as the **Acc**umulator-based **A**uthentication and Authorization model: AccA.

The core part of the architecture is built on an *authorization matrix* $\mathcal{M}$ which embeds an *authentication vector* $\mathcal{V}$. The two objects are combined and stored securely as an accumulated element of a cryptographically secure accumulator $\mathcal{Z}$, where $\mathcal{V}$ itself is also an accumulator embedded as an element in $\mathcal{M}$. The authorization matrix $\mathcal{M}$ is an $(m+1) \times n$-matrix, representing the m available functions $f_1, f_2, ..., f_m$ to be accessed in the object, and n types which are allowed to have authorization. The additional row allows for storage of the authentication vectors, see Tab. 1. The matrix $\mathcal{M}$ is represented as n vectors

Table 1: The authorization matrix with m functions (each row), n types (each column) and one corresponding authentication vector for each type.

	ID_1	ID_2	ID_3	$\cdots$	ID_n
f_1	$b_{1,1}$	$b_{2,1}$	$b_{3,1}$	$\cdots$	$b_{n,1}$
f_2	$b_{1,2}$	$b_{2,2}$	$b_{3,2}$	$\cdots$	$b_{n,2}$
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\dots$	$\vdots$
f_m	$b_{1,m}$	$b_{2,m}$	$b_{3,m}$	$\cdots$	$b_{n,m}$
	$\mathcal{V}_1$	$\mathcal{V}_2$	$\mathcal{V}_3$	$\cdots$	$\mathcal{V}_n$

$\mathcal{M}_1, ..., \mathcal{M}_n$ where each vector $\mathcal{M}_i$ corresponds to column i in $\mathcal{M}$, i.e. $\mathcal{M}_i = (b_i, \mathcal{V}_i)$, where each b_i is a bit string with corresponding bits 0 or 1 for each function, i.e. which functions ID_i is allowed and $b_i = b_{i,1} b_{i,2} \cdots b_{i,m}$. Thus, $b_{i,j}$ is the specific authorization marker for user j, i.e. for an authorization

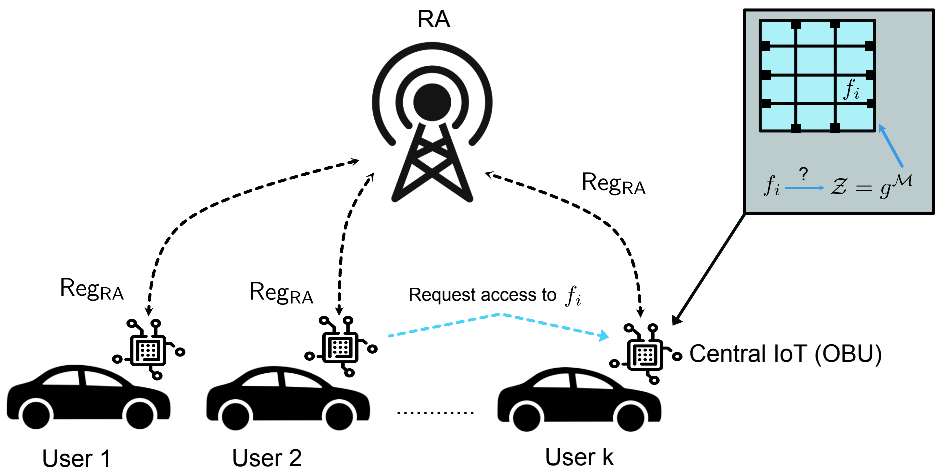

Figure 1: Ad-hoc system setting where k vehicles connect dynamically via an initial registration to a Registration Authority (RA) using the $\mathsf{Reg}_{\mathsf{RA}}$ protocol. User 2 requests access to capability f_i in vehicle k. Authentication and authorization is through the accumulator $\mathcal{Z}$ that executes in the IoT-capable Onboard Unit (OBU) in the vehicle, integrating to multiple IoT- and sensory devices in the vehicle. The OBU is regarded as an *object*.

matrix with a set of functions, then $b_{i,j}$ has a bit value of 1 for each function it have access to, 0 otherwise.

Next, the embedded authentication vector $\mathcal{V}_i$ consists of accumulated signatures of each allowed participant. The authentication vector is specified in detail:

Definition 3. *Let $v_i = \{\sigma_{i,1}, \sigma_{i,2}, \ldots, \sigma_{i,n}\}$ be a set of signatures $\sigma_{i,j} = \mathcal{H}(ID_i || \alpha_j)^{\mathsf{sk}_j}$ where $\mathcal{H}$ is a secure hash function,* sk *the private key of object j, ID_i the object identifier of type ID_i and α_j a registration value. We then call the accumulator $\mathcal{V}_i = g^{v_i} = g^{\sigma_{i,1} \cdot \sigma_{i,2} \cdots \sigma_{i,n}}$ the* authentication vector *for type ID_i*

The vector and matrix accumulation relationship is as follows:

$$\mathcal{Z} = g^{\mathcal{M}} = g^{\mathcal{M}_1 \cdots \mathcal{M}_n} = g^{(b_1, g^{v_1}) \cdot (b_2, g^{v_2}) \cdots (b_n, g^{v_n})} \quad (3)$$

We note that $\mathcal{M}$ in practical terms can be written as a single vector with $2n$ elements, but we continue to refer to the set of $\mathcal{M}_i$'s for readability. In the rest of the paper we consider each b_i as integers since the bit string from its binary representation expresses the bits for access in $\mathcal{M}$.

There are two types of witnesses used for an object's $\mathcal{Z}$: *primary* witnesses ω and *secondary* witnesses w. For an object $\mathcal{OBJ}_k$, the witness ω_i proves the existence of $\mathcal{M}_i$, hence $\omega_i^{\mathcal{M}_i} = \mathcal{Z}$. This witness is used by the object internally (that handles $\mathcal{Z}$) to ensure the authentication vectors are intact. The secondary witness w_i is generated by the object itself during registration for another object $\mathcal{OBJ}_j$. Similarly as ω_i, the proof is computed as $w_i^{\sigma_{i,j}} = \mathcal{V}_i$ and w_i is used in each request to $\mathcal{OBJ}_k$ for authentication and authorization via $\mathcal{V}_i$.

We define a combined authentication and authorization protocol, using a secure accumulator $\mathcal{Z}$ with procedures Acc and Ver for accumulation and verification of an element respectively. The architecture builds on that each object $\mathcal{OBJ}_j$ has a key-pair $\mathsf{sk}_j, \mathsf{pk}_j$ and a secure storage area with an accumulator $\mathcal{Z}$ established. The NET need a trusted infrastructure party, typically found in the C-ITS environment as a RSU or similar; we denote this trusted party as the *registration authority* (RA) (European Telecommunications Standards Institute, 2021). The RA does not need to be part of all executions in the AccA architecture, except one initial interaction with all objects that would like to be part of the NET at some future point. The aim of the RA is to establish trust with the object and allow it into the NET for further interactions. Upon gaining access through the RA, objects within the network are responsible for independently granting and revoking access to their internal functions to other objects within the network. A simplified overview of the system settings where we illustrate AccA is depicted in Fig. 1. The architecture consists of four sub-protocols:

$\mathsf{Reg}_{\mathsf{RA}}(\mathcal{OBJ}_j)$: this protocol runs between an object $\mathcal{OBJ}_j$ and the RA. Any type of initial authentication can be used, specific for the NET. After establish a secure channel, the following sub-protocol runs:

1. $\mathcal{OBJ}_j$ sends its public key $\mathsf{pk}_j = g^{\mathsf{sk}_j}$ to the RA.
2. The RA responds with $\alpha_j = \left(g^{\mathsf{sk}_j}\right)^{z_j}$ where z_j is a random value generated at the RA.
3. The RA stores z_j, its inverse $\frac{1}{z_j}$ and pk_j in a registration table, and the sub-protocol ends.

$\mathsf{Reg}_{\mathsf{OBJ}}(\mathcal{OBJ}_1, \mathcal{OBJ}_2)$: this protocol runs when $\mathcal{OBJ}_1$ need to establish an authorization in $\mathcal{OBJ}_2$. Both objects needs to be registered to the RA as a prerequisite.

1. $\mathcal{OBJ}_1$ sends a request $\rho_{register} = (\sigma_{i,1} = \mathcal{H}(\mathcal{OBJ}_1 || \alpha_1)^{\mathsf{sk}_1}, \alpha_1, f_i)$
2. $\mathcal{OBJ}_2$ sends α_1 to the RA after receiving $\rho_{register}$ from $\mathcal{OBJ}_1$.
3. The RA respond with $\beta_1 = g^{\frac{\mathsf{sk}_{RA}}{z_1}}$.
4. $\mathcal{OBJ}_2$ compute:

$$e(\alpha_1, \beta_1) \stackrel{?}{=} e(\mathsf{pk}_1, \mathsf{pk}_{RA}) \quad (4)$$

that verifies $\mathcal{OBJ}_1$ is registered in the RA, thus $\mathcal{OBJ}_2$ creates (or updates) a binary string $b_{i,1}$ that corresponds access to f_i for type 1 of $\mathcal{OBJ}_1$. We note that an object may register several types, which is determined by the identity in $\sigma_{i,1}$.

5. $\mathcal{OBJ}_2$ accumulates $\sigma_{i,1}$ into $\mathcal{Z}$ and in particular $\mathcal{V}_1$ using the Acc procedure, which in turn output corresponding witnesses ω_1 and w_i, respectively. w_i is sent back to $\mathcal{OBJ}_1$ and ω_1 is stored locally in $\mathcal{OBJ}_2$.

$\mathsf{Auth}(\mathcal{OBJ}_1, \mathcal{OBJ}_2, f_i)$: this protocol runs between two registered objects where $\mathcal{OBJ}_1$ seek access to f_i in $\mathcal{OBJ}_2$. $\mathcal{OBJ}_1$ sends a request $\rho_{access} = (h_i = \mathcal{H}(w_i)^{\mathsf{sk}_1}, w_i, g^{z_1})$. We note that g^{z_1} can only be computed by $\mathcal{OBJ}_1$ since

$$(\alpha_j)^{\frac{1}{\mathsf{sk}_1}} = g^{\frac{\mathsf{sk}_1 z_1}{\mathsf{sk}_1}} = g^{z_1} \quad (5)$$

where the inverse $\frac{1}{\mathsf{sk}_1}$ can only be computed by the holder of the secret key sk_1, i.e. $\mathcal{OBJ}_1$. The request is handled as follows:

1. $\mathcal{OBJ}_2$ verifies the BLS signature h_i:

$$e(h_i, g) \stackrel{?}{=} e(\mathcal{H}(w_i), \mathsf{pk}_j). \quad (6)$$

2. $\mathcal{OBJ}_2$ verifies that the requester is the correctly registered object:

$$e(g^{z_1}, \beta_1) \stackrel{?}{=} e(g, \mathsf{pk}_1) \quad (7)$$

3. If successfully verified, $\mathcal{OBJ}_2$ performs two witness proofs: ω_1 for the internal check of the validity of the authorization matrix, and then for the authentication vector $\mathcal{V}_1$ verifying the witness w_i, i.e. authenticating and authorizing $\mathcal{OBJ}_1$ for f_i.

$\mathsf{Rev}(\mathcal{OBJ}_1, \mathcal{OBJ}_2, f_i)$: this protocol revokes access of f_i in $\mathcal{OBJ}_2$ for $\mathcal{OBJ}_1$. In all its simplicity, $\mathcal{OBJ}_2$ runs the accumulator procedure Del to remove $\sigma_{i,1}$.

The usual C-ITS setup utilizes PKI for the fundamental trust management, using central authorities and certificate issuers (Hammi et al., 2022). However, in our proposed model, the authorization part is handled dynamically within the NET by each $\mathcal{OBJ}$. The reason is that each $\mathcal{OBJ}$ only grants access to its own capabilities and not on behalf of other objects, given the underlying trust that the RA provided via the registration. Yet, that trust is verified in every authorization registration. Therefore, the RA is only used initially for registration purposes, and all future authentication and authorization is managed by the objects themselves.

4 SECURITY ANALYSIS

Our security analysis focuses on the threat model outlined in Sec. 2.2. In forthcoming proofs we denote that an element x is excluded from a set X by $X \setminus \{x\}$.

Theorem 1. *The* AccA *architecture procedures* $\mathsf{Reg}_{\mathsf{RA}}$ *correctly register an object* $\mathcal{OBJ}_j$ *to the RA, and* $\mathsf{Reg}_{\mathsf{OBJ}}$ *correctly registers an object* $\mathcal{OBJ}_j$ *into* $\mathcal{OBJ}_k$ *for function* f_i.

Proof. Let $\mathcal{OBJ}_j$ and the RA generate their key-pairs $\mathsf{sk}_j, \mathsf{pk}_j$ and $\mathsf{sk}_{RA}, \mathsf{pk}_{RA}$ respectively. After the first exchange, $\mathcal{OBJ}_j$ stores $\alpha_j = g^{\mathsf{sk}_j z_j}$ and the RA stores z_j and g^{sk_j}. For object $\mathcal{OBJ}_k$ to verify that the registration is completed and that α_j is correct, RA returns $\beta_j = g^{\frac{\mathsf{sk}_{RA}}{z_j}}$ to $\mathcal{OBJ}_k$, then it can check that:

$$e(\alpha_j, \beta_j) = e\left(g^{\mathsf{sk}_j z_j}, g^{\frac{\mathsf{sk}_{RA}}{z_j}}\right) \quad (8)$$

$$= e\left((g^{\mathsf{sk}_j})^{z_j}, (g^{\mathsf{sk}_{RA}})^{\frac{1}{z_j}}\right) \quad (9)$$

$$= e\left(g^{\mathsf{sk}_j}, (g^{\mathsf{sk}_{RA}})^{\frac{z_j}{z_j}}\right) \quad (10)$$

$$= e\left(\mathsf{pk}_j, \mathsf{pk}_{RA}\right). \quad (11)$$

For $\mathcal{OBJ}_j$ to register access to f_i in $\mathcal{OBJ}_k$, after successful verification as described above, it run $\mathsf{Reg}_{\mathsf{OBJ}}$ for accumulation. If $\mathcal{OBJ}_k$ decide to allow for f_i then the witness w_i is sent back to $\mathcal{OBJ}_j$ as follows: since $\sigma_{i,j} = \mathcal{H}(ID_j || \alpha_j)^{\mathsf{sk}_j}$ and is accumulated as $\mathsf{Acc}(\sigma_{i,j}) = (g^{v_i})^{\sigma_{i,j}}$, the returning witness is $w_i = g^{v_i \setminus \{\sigma_{i,j}\}}$. From Def. 2 we see that it holds by definition. Hence, we conclude that $\mathsf{Reg}_{\mathsf{RA}}$ and $\mathsf{Reg}_{\mathsf{OBJ}}$ register and verifies correctly. □

We also prove the correctness of granting access to a valid object in the model:

Theorem 2. *An object $\mathcal{OBJ}_j$ is successfully granted function f_i in object $\mathcal{OBJ}_k$ if there is a valid authentication vector $\mathcal{V}$, and likewise, is not granted access to f_i if $\mathcal{OBJ}_i$ have no authentication vector established in $\mathcal{OBJ}_k$.*

Proof. $\mathcal{OBJ}_k$ receives $\rho_{access} = (h_i = \mathcal{H}(w_i)^{\mathsf{sk}_j}, w_i, g^{z_j})$ and runs the standard BLS signature verification as in Eq.6 and Eq.7. These verifies correctly since:

$$e(h_i, g) = e(\mathcal{H}(w_i)^{\mathsf{sk}_j}, g) \tag{12}$$

$$= e(\mathcal{H}(w_i), g^{\mathsf{sk}_j}) \tag{13}$$

$$= e(\mathcal{H}(w_i), \mathsf{pk}_j) \tag{14}$$

and

$$e(g^{z_j}, \beta_1) = e\left(g^{z_j}, g^{\frac{\mathsf{sk}_j}{z_j}}\right) = e\left(g, g^{\frac{\mathsf{sk}_j}{z_j}}\right)^{z_j} \tag{15}$$

$$= e\left(g, g^{\frac{z_j \mathsf{sk}_j}{z_j}}\right) = e(g, g^{\mathsf{sk}_j}) \tag{16}$$

$$= e(g, \mathsf{pk}_j). \tag{17}$$

If the verification is successful, the double witness verification with ω_j and w_i, using Ver from Def. 2 runs. Assume $\mathcal{OBJ}_k$ does not have a valid registration in $\mathcal{M}_j$, then the internal verification

$$\omega_j^{\mathcal{M}_j} = \left(g^{\mathcal{M}\setminus\{\mathcal{M}_j\}}\right)^{\mathcal{M}_j} = g^{\mathcal{M}_1 \cdots \mathcal{M}_j \cdots \mathcal{M}_n} = \mathcal{Z} \tag{18}$$

will not hold, hence the protocol will abort. This implies that $\mathcal{OBJ}_j$ must be registered in $\mathcal{OBJ}_k$. Next, assume that $\mathcal{V}_j$ does not contain $\sigma_{i,j}$, then the verification

$$w_j^{\sigma_{i,j}} = \left(g^{v_j\setminus\{\sigma_{i,j}\}}\right)^{\sigma_{i,j}} = g^{\sigma_{1,j} \cdots \sigma_{i,j} \cdots \sigma_{m,j}} = \mathcal{V}_j \tag{19}$$

will not hold, hence not give access to $\mathcal{OBJ}_j$. We recall that $v_j = \sigma_{1,j} \cdots \sigma_{m,j}$ such that $g^{v_j} = \mathcal{V}_j$.

To conclude, the protocol have ensured that $\mathcal{OBJ}_j$ is indeed the correct witness holder of w_j using the provably secure BLS scheme, it will only continue the protocol if $\mathcal{OBJ}_k$ internally verify that $\mathcal{M}_j$ is intact, and finally grant access to f_i only if w_i is a valid proof of the access signature $\sigma_{i,j}$. □

Theorem 3. *The proposed architecture is secure in the Exp_{A_1} setting.*

Proof. Let $\mathcal{A}$ be an adversary registered to the RA and previously registered to $\mathcal{OBJ}_k$ for a function f_p. $\mathcal{A}$ seeks access to f_i which is not part of the binary string b_A that represents $\mathcal{A}$'s current access in $\mathcal{OBJ}_k$. We consider two cases:

1. $\mathcal{A}$ tries to forge a witness. Let $\sigma_{p,A}$ be the signature registered for $\mathcal{A}$. Assume $\mathcal{A}$ computes w_A which is a forgery of w_i. When $\mathcal{OBJ}_k$ check the membership proof, it means that

$$w_A^{\sigma_{p,A}} = \mathcal{V}_A \tag{20}$$

should hold if successfully forged. But since $w_A = g^{v_A\setminus\{\sigma_{p,A}\}}$ it means that the adversary needs to compute all remaining signatures $\sigma_{i,j}$ in $\mathcal{OBJ}_k$'s accumulator, hence need to either forge the signatures or steal the secret key sk_j for each $\mathcal{OBJ}_j$. BLS signatures are provably secure against forgery (Boneh et al., 2001). Extracting the signatures from an existing valid witness w_q corresponding to some other function f_q, would break the security of the strong RSA assumption (Benaloh and de Mare, 1994). Therefore, even if $\mathcal{A}$ knows that the same signatures are accumulated in w_p as in w_i, it is computationally infeasible to retrieve the signatures. Stealing the secret keys would imply compromising the object's secure storage which would require a stronger adversary. Thus, forging a witness successfully is negligible. Moreover, forging the required signature $\sigma_{i,A}$ during registration is therefore also infeasible due to the security of BLS.
2. $\mathcal{A}$ tries to use a previously used witness w_{i^*} that authorized f_i but is now revoked. This case is trivial since a revocation of $\sigma_{i,A}$ means that the entry is completely deleted from $\mathcal{V}_A$, therefore $w_{i^*}^{\sigma_{i,A}} \neq \mathcal{V}_A$ regardless of the value of $\sigma_{i,A}$.

We note that manipulating b_A is not possible since it is generated in $\mathcal{OBJ}_k$ and totally dependent on what function f_i was granted during $\mathsf{Reg}_{\mathsf{OBJ}}$. To summarize, we conclude that forging a witness w_A would break the accumulator scheme in Def. 2, hence breaking the strong RSA-assumption (Def. 1). Mounting a replay-attack using a revoke witness would break the correctness of the scheme, proven valid in Thm. 2. □

A registered object, after the $\mathsf{Reg}_{\mathsf{RA}}$ protocol, has a possibility to maliciously register for a function in another object during $\mathsf{Reg}_{\mathsf{OBJ}}$. However, since we assume a dynamic, non-centralized environment NET, where each object determines to whom a registration can be granted, mitigation for such unauthorized registrations is not in scope for this paper.

Theorem 4. *The proposed architecture is secure in the Exp_{A_2} setting.*

Proof. Let $\mathcal{A}$ be an adversary not registered to the same NET as $\mathcal{OBJ}_k$, but with the ability to send messages to any target object in any network. In Exp_{A_2}

the adversary seek access to f_i in $\mathcal{OBJ}_k$ which is a registered object, hence $\mathcal{A}$ has two options:

1. Mount an unauthorized registration to $\mathcal{OBJ}_k$. Assume $\mathcal{A}$ has no registered values in the RA. This case requires $\mathcal{A}$ to run $\mathsf{Reg}_{\mathsf{OBJ}}$, i.e. to forge a request message $\rho_{register}$. Since $\mathcal{A}$ is not previously registered to the RA, it must generate a forged $\alpha_A = g^{sk_A z_A}$ for some z_A. Assume $\mathcal{A}$ generates a forgery $\rho^*_{register} = (\sigma_{i,A}, \alpha_A, f_i)$. However, when running $\mathsf{Reg}_{\mathsf{OBJ}}$, α_A is validated against the RA to receive β_A, but since there is no z_A in RA the request is rejected and no β_A exists. Therefore, regardless of what type of forgery $\rho^*_{register}$ contains, an unauthorized registration in $\mathcal{OBJ}_k$ implies the storage of z_A in RA which contradicts our assumption.
2. Forge a request ρ_{access}. Assume $\mathcal{A}$ has no registered values in the RA. $\mathcal{A}$ tries to forge $\rho_{access} = (h_A, w_i, g^{z_A})$, i.e. it must generate three components. We consider each case-by-case:
   (a) Generate w_i : from Thm. 3 we know that a witness is not possible to forge or extract.
   (b) Generate h_A : since $h_A = \mathcal{H}(w_i)^{sk_A}$ it must successfully forge w_i, and as concluded above this is not possible.
   (c) Generate g^{z_A} : as in case 1 above, there is no z_A, hence no β_A in $\mathcal{OBJ}_k$, therefore it does not matter what value g^{z_A} will have, the protocol will abort in any case.

   Therefore, none of the components are possible to generate for $\mathcal{A}$, since any successful forgery would contradict the initial assumption of $\mathcal{A}$ not having any values registered in the RA.

These two cases then conclude that AccA is secure under Exp_{A_2}. □

5 PROOF OF CONCEPT

We implemented the main parts of the AccA architecture, and conducted a set of experiments to investigate the performance. A Python wrapper of the MCL library (Mitsunari, 2019) was used with a type A curve, BLS12_381. The *hashAndMapTo* function provided in the MCL-library was used for hash function $\mathcal{H}$ when computing signatures, but also to convert accumulator elements. $\mathcal{H}$ maps to a group element in $\mathbb{G}$. Since the accumulator $\mathcal{Z}$ accumulates both integers and accumulators $\mathcal{V} \in \mathbb{G}$, we use the *hashAndMapTo* conversion for each $\mathcal{M}_i$:

```
m1 = [b_1, v_1]
m1hash = G1.hashAndMapTo(bytes(m1))
```

This minor implementation detail costs on average 0.0930 ms each time a new $\mathcal{M}_i$ needs to be added or updated. Since a NET is assumed to be short-lived and highly dynamic, the overhead is considered negligible (but included in the performance tests). All tests were executed 1000 times, and the average timings are noted in Tab. 2. Tests run on an Intel Core i5, 2.7GHz platform. As shown in Tab. 2, the Auth protocol is most expensive, naturally due to all verification procedures. We compared our results to the performance analysis made by Ometov et al. where several IoT devices were tested with a set of cryptographic primitives (Ometov et al., 2016). With a simplified comparison, we note that a pairing and a curve point multiplication, on an Intel Edison, 500 MHz Dual-Core, takes 580 ms and 0.1 ms, respectively. Hence our protocol for Auth in $\mathcal{OBJ}_k$ would take approximately 1160 ms. in the Intel Edison IoT device, if we adjust pessimistically that accumulator verification is at least as costly as as curve point multiplication. Similarly, for $\mathsf{Reg}_{\mathsf{OBJ}}$ it would take approximately 600 ms. for the RA.

Table 2: Computational performance in each object (ms).

Protocol	$\mathcal{OBJ}_j$	RA	$\mathcal{OBJ}_k$
$\mathsf{Reg}_{\mathsf{RA}}$	-	0.0671	-
$\mathsf{Reg}_{\mathsf{OBJ}}$	0.0181	1.3905	-
Auth	-	-	2.5749
Rev	-	-	0.0717

6 CONCLUSION

We have shown the feasibility and efficiency of AccA in terms of computational performance, and proven it secure against two different types of attacks. AccA can be used for autonomous IoT in short-lived and decentralized environments where only an initial registration is needed to a trusted third party. For future work we propose further investigation in how to reduce the interactive part of the protocols for efficiency reasons.

REFERENCES

Benaloh, J. and de Mare, M. (1994). One-way accumulators: A decentralized alternative to digital signatures. In Helleseth, T., editor, *Advances in Cryptology — EUROCRYPT '93*, pages 274–285, Berlin, Heidelberg. Springer Berlin Heidelberg.

Boneh, D., Lynn, B., and Shacham, H. (2001). Short signatures from the weil pairing. In *Advances in Cryptology–ASIACRYPT '01, LNCS*, pages 514–532. Springer.

Chahal, R. K., Kumar, N., and Batra, S. (2020). Trust management in social internet of things: A taxonomy, open issues, and challenges. *Computer Communications*, 150:13–46.

European Telecommunications Standards Institute (2014). ETSI EN 302 637-2: Intelligent Transport Systems (ITS); Vehicular Communications; Basic Set of Applications; Part 2: Specification of Cooperative Awareness Basic Service. https://www.etsi.org/standards. Accessed: 2022-05-15.

European Telecommunications Standards Institute (2021). ETSI TS 102 940 V2.1.1: Intelligent Transport Systems (ITS); Security; ITS communications security architecture and security management; Release 2. https://www.etsi.org/standards. Accessed: 2022-11-18.

Förster, D., Kargl, F., and Löhr, H. (2014). Puca: A pseudonym scheme with user-controlled anonymity for vehicular ad-hoc networks (vanet). In *2014 IEEE Vehicular Networking Conference (VNC)*, pages 25–32.

Galego, N. M. C. and Pascoal, R. M. (2022). Cybersecurity in smart cities: Technology and data security in intelligent transport systems. In Mesquita, A., Abreu, A., and Carvalho, J. V., editors, *Perspectives and Trends in Education and Technology*, pages 17–33, Singapore. Springer Singapore.

Gangolli, A., Mahmoud, Q. H., and Azim, A. (2022). A systematic review of fault injection attacks on iot systems. *Electronics*, 11(13).

Gupta, M., Bhatt, S., Alshehri, A. H., and Sandhu, R. (2022). *Authorization Frameworks for Smart and Connected Ecosystems*, pages 39–61. Springer International Publishing, Cham.

Hammi, B., Monteuuis, J.-P., and Petit, J. (2022). Pkis in c-its: Security functions, architectures and projects: A survey. *Vehicular Communications*, 38:100531.

Heng, X., Qin, S., Xiao, Y., Wang, J., Tao, Y., and Zhang, R. (2022). A strong secure v2i authentication scheme from pki and accumulator. In *2022 2nd International Conference on Consumer Electronics and Computer Engineering (ICCECE)*, pages 98–103.

Khan, A., Ahmad, A., Ahmed, M., Sessa, J., and Anisetti, M. (2022). Authorization schemes for internet of things: requirements, weaknesses, future challenges and trends. *Complex & Intelligent Systems*, 8(5):3919–3941.

Khan, W. Z., Hakak, S., Khan, M. K., et al. (2020). Trust management in social internet of things: Architectures, recent advancements, and future challenges. *IEEE Internet of Things Journal*, 8(10):7768–7788.

Lauinger, J., Ernstberger, J., Regnath, E., Hamad, M., and Steinhorst, S. (2021). A-poa: Anonymous proof of authorization for decentralized identity management. In *2021 IEEE International Conference on Blockchain and Cryptocurrency (ICBC)*, pages 1–9.

Mitsunari, S. (2019). "MCL cryptolibrary". https://github.com/herumi/mcl.

Ometov, A., Masek, P., Malina, L., Florea, R., Hosek, J., Andreev, S., Hajny, J., Niutanen, J., and Koucheryavy, Y. (2016). Feasibility characterization of cryptographic primitives for constrained (wearable) iot devices. In *2016 IEEE International Conference on Pervasive Computing and Communication Workshops (PerCom Workshops)*, pages 1–6.

Polychronou, N.-F., Thevenon, P.-H., Puys, M., and Beroulle, V. (2021). A comprehensive survey of attacks without physical access targeting hardware vulnerabilities in iot/iiot devices, and their detection mechanisms. *ACM Trans. Des. Autom. Electron. Syst.*, 27(1).

Ravidas, S., Lekidis, A., Paci, F., and Zannone, N. (2019). Access control in internet-of-things: A survey. *Journal of Network and Computer Applications*, 144:79–101.

Sadhu, P. K., Yanambaka, V. P., and Abdelgawad, A. (2022). Internet of things: Security and solutions survey. *Sensors*, 22(19).

Salin, H. and Fokin, D. (2022). An authenticated accumulator scheme for secure master key access in microservice architectures. In Bastieri, D., Wills, G. B., Kacsuk, P., and Chang, V., editors, *Proceedings of the 7th International Conference on Internet of Things, Big Data and Security, IoTBDS 2022, Online Streaming, April 22-24, 2022*, pages 119–126. SCITEPRESS.

Salin, H., Fokin, D., and Johansson, A. (2021). Authenticated multi-proxy accumulation schemes for delegated membership proofs. In Luo, B., Mosbah, M., Cuppens, F., Othmane, L. B., Cuppens, N., and Kallel, S., editors, *Risks and Security of Internet and Systems - 16th International Conference, CRiSIS 2021, Virtual Event, Ames, USA, November 12-13, 2021, Revised Selected Papers*, volume 13204 of *Lecture Notes in Computer Science*, pages 162–171. Springer.

Shahab, S., Agarwal, P., Mufti, T., and Obaid, A. J. (2022). Siot (social internet of things): A review. In Fong, S., Dey, N., and Joshi, A., editors, *ICT Analysis and Applications*, pages 289–297, Singapore. Springer Nature Singapore.

Sharma, A., Pilli, E. S., Mazumdar, A. P., and Gera, P. (2020). Towards trustworthy internet of things: A survey on trust management applications and schemes. *Computer Communications*, 160:475–493.

Valle, F., Cooney, M., Mikhaylov, K., and Vinel, A. (2021). The integration of uavs to the c-its stack. In *2021 IEEE 29th International Conference on Network Protocols (ICNP)*, pages 1–6.

Zeddini, B., Maachaoui, M., and Inedjaren, Y. (2022). Security threats in intelligent transportation systems and their risk levels. *Risks*, 10(5).

Zuo, X., Li, L., Luo, S., Peng, H., Yang, Y., and Gong, L. (2021). Privacy-preserving verifiable graph intersection scheme with cryptographic accumulators in social networks. *IEEE Internet of Things Journal*, 8(6):4590–4603.