Device Identification and Personal Data Attestation in Networks

Clémentine Gritti, Melek Önen, Refik Molva, Willy Susilo, Thomas Plantard

Device Identification and Personal Data Attestation in Networks

Clémentine Gritti[1]*, Melek Önen[2], Refik Molva[2], Willy Susilo[3], and Thomas Plantard[3]
[1]NTNU, Norway
clementine.gritti@ntnu.no
[2]Eurecom, France
melek.onen@eurecom.fr, refik.molva@eurecom.fr
[3]University of Wollongong, Australia
wsusilo@uow.edu.au, thomaspl@uow.edu.au

Abstract

A powerful world connecting digital and physical environments is promised through the Internet of Things (IoT). However, because of the heterogeneous nature of devices and of the diversity of their provenance, security and privacy vulnerabilities threaten IoT-based implementations. Moreover, constrained resources from devices bring technical challenges, compelling protocols to be as lightweight as possible. To overcome such problems, we propose an efficient solution for identity and data management in IoT. Similarly to Gritti et al.'s approach, a secure bootstrap is first processed to enable a reliable authentication of devices in a local network, and then, a message attestation phase is executed to allow authentication of personal messages of devices. While devices are limited to pre-determined common messages in Gritti et al.'s solution, they can authenticate their own personal messages in our paper. We ensure that our solution is suitable in IoT settings by proving it secure and privacy-preserving as well as satisfying operational requirements. In addition, we provide benchmarking results on both the scheme from Gritti et al.'s scheme and our scheme.

Keywords: Internet of Things, Identity-Based Cryptography, Aggregate Signature

1 Introduction

Computation and communication technologies have been evolving with the emergence of the Internet of Things (IoT). Areas such as of business and industry, highly benefit from IoT by melting digital and physical environments. Nevertheless, the IoT technology brings new challenges for identification and data management. The nature of IoT makes devices be heterogeneous in terms of origin and functionality. Indeed, vendors manufacturing devices are manifold, contrary to the ones for mobile and wireless devices. Moreover, the number of operators exploiting IoT devices is huge. Hence, such heterogeneity threatens expected guarantees on security and privacy, and solutions developed for IoT networks should thus provide reliable authentication of devices. Other problems appear regarding the technical features of devices due to their constrained resources in terms of CPU, memory, bandwidth and storage. Therefore, secure and privacy-preserving IoT-based solutions must be as lightweight as possible according to the computational and storage limitations of the devices. Eventually, an authentication protocol designed for IoT settings must satisfy the following points: security and privacy on one side and efficiency and practicality on the other side, because of the diversity of parties involved into the conception and exploitation of devices as well as the limitations of devices' resources. By considering the above challenges in dynamic and self-organized IoT networks, we aim to create a reliable and effective protocol for identity and data authentication.

Journal of Wireless Mobile Networks, Ubiquitous Computing, and Dependable Applications (JoWUA), volume: 9, number: 4 (December), pp. 1-26

*Corresponding author: Elektro B, Gloshaugen, Department of Information Security and Communication Technology, NTNU, Trondheim, Norway

One might mention Public Key Infrastructures (PKIs) for device registration and setup process. Nevertheless, global PKIs are not suitable in IoT networks due to the devices' provenance diversity and lack of governance. In addition, using certificates overly costs while devices have limited capacities. Therefore, a secure and privacy-preserving solution for identity and data management that is applicable in IoT networks should take an alternative to PKIs, based on two phases. A first one, that is device identity authentication, enables a secure bootstrap and light setup process. A second one, that is device response authentication, allows a reliable message attestation. We suggest to solve the origin diversity and lack of governance problems by focusing on device identity and data management locally. Instead of considering the problems in a global (worldwide) space, we demarcate them in confined IoT networks, set as smart homes, smart vehicles, smart wearables, smart bodies, etc.

We design a public-key cryptographic scheme by embedding the notion of aggregate signature in an identity-based setting. The former enables message attestation by aggregating personal responses from devices and verifying the aggregate response. Identity-based cryptography permits secure bootstrap without using bulky certificates. One might think that a simple combination of these two primitives would be enough for our purposes. However, this combination should be improved in order to satisfy the technical requirements from devices' constrained resources. In order to illustrate the practicality of our scheme in IoT environments, we implement it and calculate time and memory consumption. Since no experimental results were provided, we do the same for a similar scheme presented at SAC 2018 [14], where the main difference with ours comes from techniques used for message attestation (authors in [14] used multi-signatures while we use aggregate signatures).

In Section 2, we detail the challenges and related issues brought by device identity and data authentication in networks. In Section 3, we develop the main features that should satisfy our solution in order to solve the challenges mentioned in the previous section. In Section 4, the definition of our solution enabling secure bootstrap and message attestation in IoT networks is given. In Section 5, the construction and security analysis of our solution are provided. In Section 7, we present the benchmarking results of our solution and of the scheme in [14]. In Section 8, we recall the existing works that focus on similar challenges. We conclude our research work in Section 9.

2 Statement of the Problem

Problems related to device identity management in IoT impact on security and privacy. Device identification and authentication challenges have already been studied. Nevertheless, considering these challenges in the context of IoT is relatively new, and unfortunately, existing protocols do not suit such context. The novelty comes from the heterogeneity of devices regarding their provenance and functionality, as well as their constantly increasing number, making the task of assigning unique identity to devices difficult. Contrary to mobile networks that involve countable vendors and operators and thus make authentication process more easily trusted, IoT networks imply a too large number of such parties. The worst realistic scenario engages devices with unknown provenance, communicating with other devices without initial registration, making malicious intrusions be likely to happen.

Public Key Infrastructures (PKIs) require the use of certificates that appear as an answer for reliable identity authentication. Nevertheless, practical limitations have been encountered in some PKI-based applications, such as for the Secure Border Gateway Protocol [20, 9]. Because of the dynamicity and self-organization of IoT networks, a solution relying on a global PKI, where certification authorities could not be recognized and accepted, seems difficult to implement. Moreover, since certificates are voluminous compared to the devices' restricted computation, communication and storage resources, implementing a PKI in IoT networks would bring an expensive burden. The size of certificates is not the only issue; identity naming for the generation of certificates would be arduous if done globally (i.e.

worldwide). Such task has already been noticed as cumbersome in traditional X.509 PKIs since defining proper distinct names for numerous parties increases the infrastructure complexity. Instead, device identity attribution should remain meaningful in IoT networks.

In this context, Gritti et al. [14] propose to tackle the identity authentication problem in IoT networks at a local level by defining confined networks. The authors introduce the notion of Subnet of Things such that smart devices of a confined network communicate among them and rely on a central node linking them to the Internet. The IoT devices and central node, exchanging information locally, thus establish a subnet in the global setting. Thus, similarly to [14], we solve the problem related to identity and data management by considering a local setting, i.e. in a confined network. The identity authentication phase relies on the one introduced in [14]. This phase locally attributes unique identities to devices in a confined network, enabling secure bootstrap. Thereafter, the devices and central node start communicating together. The message attestation phase is done by verifying the aggregate response generated by all the devices in the confined network, avoiding to check individual responses one by one. Contrary to [14] that let devices with limited choices of individual responses, we permit them to personalize their responses in our solution. Aggregation is done on all their personal responses, and verification is executed on the resulting response that should be compact and complete regarding all the participating devices in the confined network. Unfortunately, no experimental results are given in [14] to show the feasibility of the solution in IoT environments. We thus give benchmarking results of the scheme in [14], along with the ones of our scheme, and discuss on them regarding IoT setting requirements.

3 Idea of the Solution

We present a solution to overcome the issue of identity and message authentication in confined IoT networks. Similarly to [14], we consider a local space instead of a global one, solving the problem of accurately identifying devices. Identity authentication is done by attributing a specific role to each device in a given network. Hence, the identification of devices is not global; it rather depends on the network in which the device is located. Then, message authentication is executed on behalf of the network, rather than on behalf of devices taken individually, by considering devices as a group in their network. The verification is performed on the aggregation of their responses instead of on their individual ones. Moreover, devices can personalize their responses, rather than choosing pre-determined common messages as in [14], allowing unrestrained communication among devices. We illustrate our idea with an applicative example in medicine and health care, such that IoT networks could enable effective post-operative or intensive care: by disposing sensors on medical machines or patient bodies, we enable to monitor chronically ill patients or elderly. Another example comes from preventive maintenance. By embedding sensors with control functions into places that cables cannot reach (for instance, on tires for pressure monitoring), we permit real machine surveillance.

We enable devices to be uniquely identified according to their attributes including localization (i.e. a confined network) and role (i.e. functionality in this network). By doing so, we enhance secure bootstrap in IoT networks. Identity-Based Signatures (IBS) [21] appear as a way to permit secure bootstrap in a local space and avoid the use of onerous certificates. In an IBS scheme, the private key is linked to a public key that is simply the device's identity. However, IBS only allows for the verification of individual responses instead of the verification of an aggregate one. This incurs that, in an IBS scheme, each device's response will be separately checked and validated, adding extra burden in the message attestation process. Aggregate Signatures (AS) [7, 18] afford message attestation on behalf of a group of devices by aggregating their individual personalized signatures into a global one, such that the latter must be as compact as wished (i.e. not depending on the number of signing devices). Existing AS-based solutions rely on PKIs, and hence identity authentication requires certification, making bootstrap

inefficient in the context of IoT. To achieve secure and privacy-preserving identity and data management in IoT networks, we combine the primitives of IBS and AS. In addition, the organization of our local space appears to be hierarchical: the confined network could play the role of a parent while devices would be siblings. We hence present a 2-level hierarchical Identity-Based Aggregate Signature (2-IBAS) scheme where identification of devices follows their localization in a confined network and their role within this network, and exchanged data attestation relies on signing and verifying processes. We also provide performance analysis based on time and memory consumption of our solution and of the one in [14] as an illustration of the practicality of such schemes in IoT settings.

4 Protcol for Secure Bootstrap and Aggregate Response Attestation

4.1 Overview of the Solution

Three parties participate into the secure bootstrap and aggregate response validation protocol:
Confined Network: A network should be structured as confined, with a limited number of distinct devices. This network can be described as a local space in the whole Internet. The local IoT network is given an identity that is used to generate its private key.
Devices: Embedded into a confined network, the devices receive identities that depend on their localization (their network) and their role within the network. These identities are used to generate the signing keys of the devices. Devices are responsible for computing and transmitting personal responses one after the other. Each response is generated according to the responses of the previous devices as well as their personal message to be attested. An aggregate response is then obtained on behalf of all the participating devices and sent to a verifier that collects and checks the validity of such response.
Response Verifier: The task of the verifier is to collect and check the aggregate response generated by the devices located in a same network. The verification is done on the aggregate response and the to-be-attested personal messages from all the devices. The verification is a public process, hence the verifier is not a party designated beforehand.

Three phases compose the secure bootstrap and aggregate response validation protocol:
Network and Device Setup: The confined network is assigned an identity and receives a relevant private key (generated by an off-line trusted third party). Devices are attributed local and unique identities in a secure manner, and are given the corresponding signing keys by their common network.
Aggregate Response Generation: The devices must sign their personal messages in a sequential manner. Their messages are all distinct such that we link one message to one device. Messages are chosen regarding the verifier's request as well as the devices' state. Once all devices signed their messages, the aggregate result represents the group response of all the participating devices.

We succinctly describe the sequential signing process. The request from the verifier is received by a device in the network. The latter answers personally to this request by signing its own message with its signing key and sends the resulting signature to a subsequent device. This second device replies to the request by generating a signature on its personal message, using its signing key and the signature transmitted by the first device. The second resulting signature is forwarded to a next device. The process goes on as above, where a device needs its personal message, its signing key and the signature transmitted by a previous device to compute its own signature. The last device participating in the process in the same network generates its signature and sets it as the aggregate response. This response depends on all the signatures of the devices, rather than being a simple combination of individual responses from these devices. The aggregate response is forwarded to the response verifier as the answer to the initial request. All the personalized messages should be known by the verifier in order to accurately check the validity of

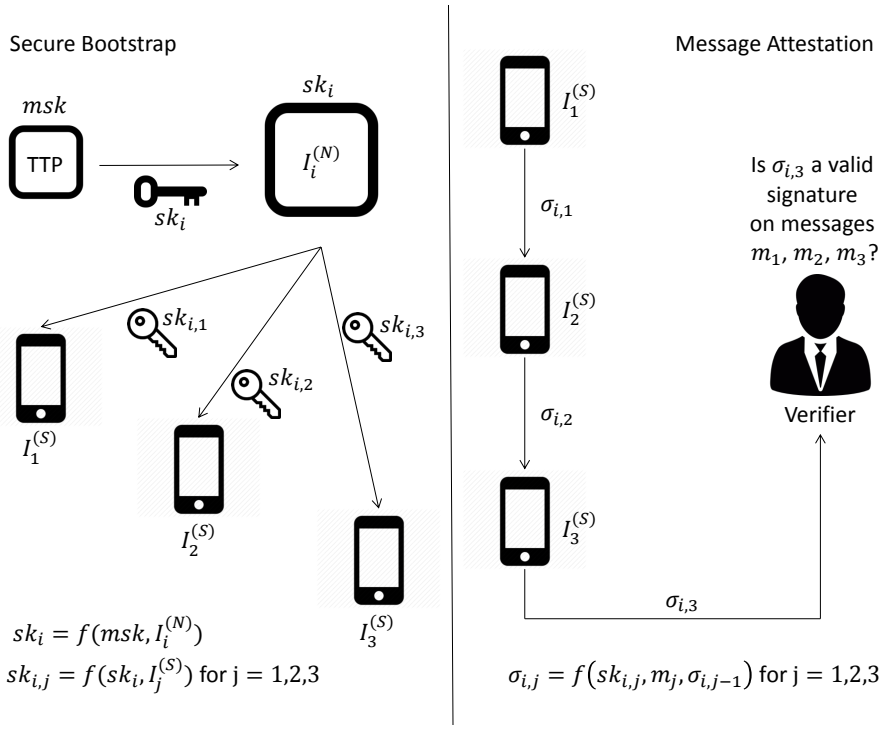


Figure 1: Protocol overview with a network, three devices and a response verifier participating.

the aggregate response. We implicitly assume that each device embeds its own message with its signature when transmitting the latter to the next device.

Aggregate Response Validation: The response verifier, who could be anyone, is given the aggregate response and checks its validity. If the response is actually valid, then the messages of the devices are well attested; otherwise, attestation fails. An aggregate signature convinces a verifier whether each device has signed a particular message.

Figure 1 illustrates the protocol where where a network, three devices and a response verifier participate. TTP refers to a trusted third party. We give more details about the keys and signature in the next section.

4.2 2-level Identity-Based Aggregate Signature Scheme

The element $I_i^{(N)}$ is the identity of the network and the element $I_j^{(D)}$ is the identity of the device referring to its role within the network. All identities and to-be-attested messages are supposed to be distinct. The 2-level Identity-Based Aggregate Signature scheme (2-IBAS) contains the following five algorithms:

Network and Device Setup:

$\mathsf{Setup}(\lambda) \rightarrow (params, msk)$ is run by a trusted third party. It takes as inputs the security parameter λ, and it outputs the public parameters *params* and the master secret key *msk*. The public parameters *params* are available to the network, devices and verifier. The master secret key *msk* is kept secret by the trusted third party.

$\mathsf{KeyGen}_1(params, msk, I_i^{(N)}) \rightarrow sk_i$ is run by a trusted third party. It takes as inputs the public parameters *params*, the master secret key *msk* and the identity $I_i^{(N)}$ of the network, and it outputs the private key sk_i

that is sent to the network.

$\mathsf{KeyGen}_2(params, sk_i, I_j^{(D)}) \rightarrow sk_{i,j}$ is run by the network in which the devices are installed. It takes as inputs the public parameters *params*, the private key sk_i of the network, and the identity $I_j^{(D)}$ of a device in this network, and it outputs the signing key $sk_{i,j}$ that is sent to the device.

Aggregate Response Generation:

$\mathsf{Sign}_j(params, sk_{i,j}, \sigma_{i,j-1}, m_j) \rightarrow \sigma_{i,j}$ is run by a device. It takes as inputs the public parameters *params*, the signing key $sk_{i,j}$ of the device, the signature $\sigma_{i,j-1}$ generated by the previous signing device with identity $I_{j-1}^{(D)}$ in the network (for $j = 1$, $\sigma_{0,i} = \perp$ since there is no previous signature), and a personal message m_j, and it outputs the signature $\sigma_{i,j}$ that is forwarded to the subsequent signing device with identity $I_{j+1}^{(D)}$ in the network.

Aggregate Response Validation:

$\mathsf{Verif}(params, I_i^{(N)}, \{I_j^{(D)}\}_{j\in[1,l]}, \sigma_{i,l}, \{m_j\}_{j\in[1,l]}) \rightarrow \{$"Accept","Reject"$\}$ is run by the response verifier. It takes as inputs the public parameters *params*, the identity $I_i^{(N)}$ of a network, the set of identities $\{I_j^{(D)}\}_{j\in[1,l]}$ of the devices in this network (where l is the number of devices located in this network), an aggregate signature $\sigma_{i,l}$ corresponding to the signature generated by the l-th (last) device, and a set of personal messages $\{m_j\}_{j\in[1,l]}$, and it outputs either "Accept" or "Reject".

Correctness

For all $(params, msk) \leftarrow \mathsf{Setup}(\lambda)$, keys $sk_i \leftarrow \mathsf{KeyGen}_1(params, msk, I_i^{(N)})$ and $sk_{i,j} \leftarrow \mathsf{KeyGen}_2(params, sk_i, I_j^{(D)})$ and personal messages m_j for $j \in [1,l]$, if the $l-1$ signatures are generated as $\sigma_{i,j} \leftarrow \mathsf{Sign}_j(params, sk_{i,j}, \sigma_{i,j-1}, m_j)$ for $j \in [1,l-1]$, and if the aggregate signature is generated as $\sigma_{i,l} \leftarrow \mathsf{Sign}_l(params, sk_{i,l}, \sigma_{i,l-1}, m_l)$, then $\mathsf{Verif}(params, I_i^{(N)}, \{I_j^{(D)}\}_{j\in[1,l]}, \sigma_{i,l}, \{m_j\}_{j\in[1,l]}) \rightarrow$ "Accept".

5 Construction and Analysis of the Security

5.1 Preliminaries

Bilinear Maps: Let G and G_T be two multiplicative cyclic groups of prime order p according to the security parameter λ. Let g be a generator of G. Let $e : G \times G \rightarrow G_T$ be a bilinear map such that:

- $\forall u, v \in G, \forall a, b \in Z_p$, $e(u^a, v^b) = e(u, v)^{ab}$ (bilinearity).
- $e(g, g) \neq 1_{G_T}$ (non-degeneracy).
- $\forall a, b \in Z_p$, $e(g^a, g^b) = e(g, g)^{ab} = e(g^b, g^a)$ (symmetry).

G is a bilinear group if the group operation in $G \times G$ and the bilinear map e are both efficiently computable.

Computational Diffie-Hellman (CDH) Assumption: Let G be a group of prime order p according to the security parameter λ. Let $a, b \in_R Z_p$ and g be a generator of G. The problem is: If an adversary $\mathscr{A}$ is given a CDH tuple (g, g^a, g^b), it remains hard to compute $g^{ab} \in G$. The CDH assumption holds if no probabilistic polynomial-time adversary $\mathscr{A}$ has non-negligible advantage in solving the CDH problem.

5.2 Construction

Gritti et al. [14] propose a 2-level Identity-Based Multi-Signature (2-IBMS) scheme to enable local secure bootstrap and message attestation in IoT. We propose to extend such scheme by using the Gentry-

Ramzan technique [12] to allow devices to sign personal messages that are all distinct instead of signing pre-selected common messages. Gentry and Ramzan [12] extend their Identity-Based Multi-Signature (IBMS) scheme into an Identity-Based Aggregate Signature (IBAS) by letting parties sign a common dummy message following the signing process of their former scheme. Then, each signer is able to aggregate their personal message's signature by aggregating their randomness and by embedding their message into the resulting signature. Our 2-level Identity-Based Aggregate Signature (2-IBAS) construction is as follows:

Network and Device Setup:

$\mathsf{Setup}(\lambda) \rightarrow (params, msk)$. Given the security parameter λ, let G, G_T be two cyclic multiplicative groups of prime order p. Let g be a generator of G and $e : G \times G \rightarrow G_T$ be a bilinear map. The trusted third party randomly chooses $\alpha, \beta \in_R Z_p$ and computes $h_1 = g^{\alpha}$ and $h_2 = g^{\beta}$. Let $H_1, H_2, H_3 : \{0,1\}^* \rightarrow G$ and $H_4 : \{0,1\}^* \rightarrow Z_p$ be four cryptographic hash functions seen as random oracles. Finally, the trusted third party sets the public parameters as $params = (p, G, G_T, e, g, h_1, h_2, H_1, H_2, H_3, H_4)$ and the master secret key as $msk = (\alpha, \beta)$.

$\mathsf{KeyGen}_1(params, msk, I_i^{(N)}) \rightarrow sk_i$. The trusted third party computes $g_i = H_1(I_i^{(N)})$ and $C_i^{(1)} = g_i^{\alpha}$, and sets $C_i^{(2)} = \beta$. It sets the private key of the network with identity $I_i^{(N)}$ as $sk_i = (C_i^{(1)}, C_i^{(2)})$.

$\mathsf{KeyGen}_2(params, sk_i, I_j^{(D)}) \rightarrow sk_{i,j}$. The network parses its private key sk_i as $(C_i^{(1)}, C_i^{(2)})$. It computes $g_{j,0} = H_2(I_j^{(D)}, 0)$, $g_{j,1} = H_2(I_j^{(D)}, 1)$, $D_{i,j}^{(1)} = C_i^{(1)} \cdot g_{j,0}^{C_i^{(2)}}$ and $D_{i,j}^{(2)} = C_i^{(1)} \cdot g_{j,1}^{C_i^{(2)}}$, and sets the signing key of the device with identity $I_j^{(D)}$ in the network with identity $I_i^{(N)}$ as $sk_{i,j} = (D_{i,j}^{(1)}, D_{i,j}^{(2)})$.

Aggregate Response Generation:

$\mathsf{Sign}_j(params, sk_{i,j}, \sigma_{i,j-1}, m_j) \rightarrow \sigma_{i,j}$. The device parses its signing key $sk_{i,j}$ as $(D_{i,j}^{(1)}, D_{i,j}^{(2)})$. For $j = 1$, the device with identity $I_1^{(D)}$ selects a string w that has never been used for other signatures. For $j > 1$, the device with identity $I_j^{(D)}$ checks that the string w has not been used for other signatures.. The device then randomly chooses $t_j \in_R Z_p$ and computes $g_w = H_3(w)$, $a_j = H_4(m_j, I_j^{(D)}, w)$ and its individual elements $B_{i,j}^{(1)} = g_w^{t_j} \cdot D_{i,j}^{(1)} \cdot (D_{i,j}^{(2)})^{a_j}$ and $B_{i,j}^{(2)} = g^{t_j}$. For $j = 1$, $\sigma_{i,0} = \perp$ and the device sets its individual signature as $\sigma_{i,1} = (B_{i,1}^{(1)}, B_{i,1}^{(2)}, w)$.

Given the signature $\sigma_{i,j-1} = (S_{i,j-1}^{(1)}, S_{i,j-1}^{(2)}, w)$ from the previous signing device in the same network, for $j > 1$, the device generates the aggregate elements:

$$
\begin{aligned}
S_{i,j}^{(1)} &= S_{i,j-1}^{(1)} \cdot B_{i,j}^{(1)} = \prod_{j'=1}^{j} B_{i,j'}^{(1)} = g_w^{\sum_{j'=1}^{j} t_{j'}} \cdot g_i^{j\alpha} \cdot \prod_{j'=1}^{j} g_{j'}^{\beta} \cdot g_i^{\alpha \sum_{j'=1}^{j} a_{j'}} \cdot \prod_{j'=1}^{j} g_{j'}^{\beta a_{j'}} \\
S_{i,j}^{(2)} &= S_{i,j-1}^{(2)} \cdot B_{i,j}^{(2)} = \prod_{j'=1}^{j} B_{i,j'}^{(2)} = g^{\sum_{j'=1}^{j} t_{j'}}
\end{aligned}
$$

and sets its signature as $\sigma_{i,j} = (S_{i,j}^{(1)}, S_{i,j}^{(2)}, w)$.

Aggregate Response Validation:

$\mathsf{Verif}(params, I_i^{(N)}, \{I_j^{(D)}\}_{j \in [1,l]}, \sigma_{i,l}, \{m_j\}_{j \in [1,l]}) \rightarrow \{$"Accept","Reject"$\}$. Given the identity $I_i^{(N)}$ of the network and the set of identities $\{I_j^{(D)}\}_{j \in [1,l]}$ of the devices within this network, the set of personal messages $\{m_j\}_{j \in [1,l]}$, and the aggregate signature $\sigma_{i,l} = (S_{i,j}^{(1)}, S_{i,j}^{(2)}, w)$, the response verifier checks whether

the following equation holds:

$$e(S_{i,l}^{(1)},g) = e(H_3(w),S_{i,l}^{(2)}) \cdot e(H_1(I_i^{(N)}),h_1^l \cdot h_1^{\sum_{j=1}^{l} a_j}) \cdot e(\prod_{j=1}^{l} H_2(I_j^{(D)},0) \cdot H_2(I_j^{(D)},1)^{a_j},h_2)$$

where $a_j = H_4(m_j,I_j^{(D)},w)$. If the above equation holds, then the response verifier outputs "Accept"; otherwise, it outputs "Reject".

Correctness

Let l be the number of devices in the network. Let $I_i^{(N)}$ be the identity of the network, and $\{I_j^{(D)}\}_{j\in[1,l]}$ be the set of identities of the devices. Let $\sigma_{i,l} = (S_{i,l}^{(1)},S_{i,l}^{(2)},w)$ be the aggregate signature and $\{m_j\}_{j\in[1,l]}$ be the set of personal distinct messages. Let $a_j = H_4(m_j,I_j^{(D)},w)$.

$$\begin{aligned}
e(S_{i,l}^{(1)},g) &= e(g_w^{\sum_{j=1}^{l} t_j} \cdot g_i^{l\alpha} \cdot \prod_{j=1}^{l} g_{j,0}^{\beta} \cdot g_i^{\alpha \sum_{j=1}^{l} a_j} \cdot \prod_{j=1}^{l} g_{j,1}^{\beta a_j},g) \\
&= e(g_w^{\sum_{j=1}^{l} t_j},g) \cdot e(g_i^{l\alpha},g) \cdot e(\prod_{j=1}^{l} g_{j,0}^{\beta},g) \cdot e(g_i^{\alpha \sum_{j=1}^{l} a_j},g) \cdot e(\prod_{j=1}^{l} g_{j,1}^{\beta a_j},g) \\
&= e(g_w,g^{\sum_{j=1}^{l} t_j}) \cdot e(g_i,g^{l\alpha}) \cdot e(g_i,g^{\alpha \sum_{j=1}^{l} a_j}) \cdot e(\prod_{j=1}^{l} g_{j,0},g^{\beta}) \cdot e(\prod_{j=1}^{l} g_{j,1}^{a_j},g^{\beta}) \\
&= e(g_w,S_{i,l}^{(2)}) \cdot e(g_i,h_1^l) \cdot e(g_i,h_1^{\sum_{j=1}^{l} a_j}) \cdot e(\prod_{j=1}^{l} g_{j,0},h_2) \cdot e(\prod_{j=1}^{l} g_{j,1}^{a_j},h_2) \\
&= e(H_3(w),S_{i,l}^{(2)}) \cdot e(H_1(I_i^{(N)}),h_1^l) \cdot e(H_1(I_i^{(N)}),h_1^{\sum_{j=1}^{l} a_j}) \\
&\quad \cdot e(\prod_{j=1}^{l} H_2(I_j^{(D)},0),h_2) \cdot e(\prod_{j=1}^{l} H_2(I_j^{(D)},1)^{a_j},h_2) \\
&= e(H_3(w),S_{i,l}^{(2)}) \cdot e(H_1(I_i^{(N)}),h_1^l \cdot h_1^{\sum_{j=1}^{l} a_j}) \cdot e(\prod_{j=1}^{l} H_2(I_j^{(D)},0) \cdot H_2(I_j^{(D)},1)^{a_j},h_2)
\end{aligned}$$

5.3 Analysis of the Security

5.3.1 Model

The security model follows the one given in [14] with the change of signature type by considering distinct personal messages instead of one common to all the signers. We show that our 2-IBAS scheme is secure against existential forgery under chosen message and chosen identity attacks. Let an adversary $\mathscr{A}$ and a challenger $\mathscr{B}$ interact. $\mathscr{A}$'s aim is to forge an aggregate signature on identities and messages of its choice. It is allowed to make queries for first order key generation and for second order key generation on all but one of these identities, as well as to choose messages used in its forgery such that the identity has not been used for another query with a different message. It can also make queries for signature generation on any identity and any message.

Setup: $\mathscr{B}$ runs the algorithm Setup on input the security parameter λ, and obtains the public parameters *params* and the master secret key *msk*. It gives *params* to $\mathscr{A}$ and keeps *msk* secret.

Adaptive Queries: $\mathscr{A}$ adaptively makes queries to $\mathscr{B}$ as follows:

- *First Order Key Generation Query:* $\mathscr{A}$ can request the private key associated with the identity $I_i^{(N)}$ of a network. $\mathscr{B}$ answers by running the algorithm KeyGen_1 and gives the resulting key sk_i to $\mathscr{A}$.
- *Second Order Key Generation Query:* $\mathscr{A}$ can request the signing key associated with the identity $I_j^{(D)}$ of a device in a network with identity $I_i^{(N)}$. $\mathscr{B}$ answers by running the algorithm KeyGen_2 with input the private key sk_i which has been output by the algorithm KeyGen_1, and gives the resulting key $sk_{i,j}$ to $\mathscr{A}$.
- *Signature Query:* $\mathscr{A}$ requests the signature associated with the identity $I_j^{(D)}$ of the device in the network with identity $I_i^{(N)}$ on a message m_j of its choice. $\mathscr{B}$ answers by running the algorithm Sign_j with input the signing key $sk_{i,j}$ which has been output by the algorithm KeyGen_2 and the previous signature $\sigma_{i,j-1}$ which has been output by the algorithm Sign_{j-1}, and gives the resulting signature $\sigma_{i,j}$ to $\mathscr{A}$. We require that the adversary has not made a query for a signature on a message m'_j with an identity $I_j^{(D)}$ already used for another query on a distinct message $m_j \neq m'_j$.

Forgery: $\mathscr{A}$ outputs an aggregate signature σ^*, an identity $I_i^{(N)*}$ and a set of identities $\{I_j^{(D)*}\}_{j\in[1,l]}$ (such that the devices with identities $I_j^{(D)*}$ are in the same network) and a set of distinct messages $\{m_j^*\}_{j\in[1,l]}$. $\mathscr{A}$ wins the game if the following conditions are true:

- The output of the verification algorithm on inputs $\{m_j^*\}_{j\in[1,l]}$, σ^*, $I_i^{(N)*}$ and $\{I_j^{(D)*}\}_{j\in[1,l]}$ is "Accept";
- There exists a network with identity $I_i^{(N)*}$ for which $\mathscr{A}$ has not made a first order key generation query and a device with identity $I_j^{(D)*}$ in this network for which $\mathscr{A}$ has not made a second order key generation query (meaning that there is at least one uncorrupted identity for each party);
- $\mathscr{A}$ has not made signature query on a signing key associated with a device with identity $I_j^{(D)*}$ in the network with identity $I_i^{(N)*}$ and the message m_j^*.

The advantage of the adversary $\mathscr{A}$ in winning the above game is defined as $Adv_{\mathscr{A}} = Pr[\mathscr{A} \text{ wins}]$ where the probability is taken over all coin tosses made by the challenger $\mathscr{B}$ and the adversary $\mathscr{A}$. We say that a 2-IBAS scheme is secure against forgery under chosen message and chosen identity attacks if there does not exist a probabilistic polynomial-time adversary $\mathscr{A}$ with non-negligible advantage $Adv_{\mathscr{A}}$.

5.3.2 Sketch of the Security Proof

We only provide some intuition on a challenger $\mathscr{B}$ being successful in solving the CDH problem while interacting with an adversary $\mathscr{A}$. We let the reader to refer to [12, 14] for more details.

An adversary $\mathscr{A}$ wishes to break the security of the 2-IBAS scheme in the random oracle model. A challenger $\mathscr{B}$ attempts to solve the CDH problem by interacting with $\mathscr{A}$. A CDH tuple (g, g^a, g^b) is given to $\mathscr{B}$. Let $h_2 = g^b$. The hash functions H_1, H_2, H_3 and H_4 are controlled by $\mathscr{B}$.
H_1 **queries:** To answer, $\mathscr{B}$ randomly chooses an element ν_i in Z_p and computes $g_i^{\alpha} = g^{\alpha\nu_i}$.
H_2 **queries:** To answer, $\mathscr{B}$ picks at random $\mu_{j,0}, \mu_{j,1} \in_R Z_p$ and computes $g_{j,0}^{\beta} = g^{b\mu_{j,0}}$ and $g_{j,1}^{\beta} = g^{b\mu_{j,1}}$. Nevertheless, $\mathscr{B}$ sometimes computes $g_{j,0} = g^{\mu_{j,0}} \cdot (g^a)^{\mu'_{j,0}}$ and $g_{j,1} = g^{\mu_{j,1}} \cdot (g^a)^{\mu'_{j,1}}$ for some elements $\mu'_{j,0}, \mu'_{j,1} \in Z_p$. Thus, in such situation, $\mathscr{B}$ is not able to answer to a second order key generation query on identity $I_j^{(D)}$. However, in the case of this identity $I_j^{(D)}$ being the target choice of $\mathscr{A}$, the forgery of the latter could help $\mathscr{B}$ to solve the CDH problem.
H_3 **queries:** To answer, $\mathscr{B}$ computes $g_w = (g^a)^{d_w}$ for a known random exponent $d_w \in Z_p$ most of the time. Nevertheless, it sometimes computes $g_w = g^{c_w}$ for another known random exponent $c_w \in Z_p$.

H_4 **queries:** To answer, $\mathscr{B}$ randomly chooses an element ξ_j and computes $a_j = g^{\xi_j}$ if it knows d_w. Otherwise, it calculates $a_j = g^{\xi}$ such that the exponent ξ is a unique value that helps to cancel out the multiple of g^{ab} in $S_{i,j}^{(1)}$.

$\mathscr{B}$ is able to reply to a signature query on identity $I_j^{(D)}$, dummy message w and message m_j by controlling the H_2, H_3 and H_4 oracles, although it cannot get the signing key linked to identity $I_j^{(D)}$. There are two cases:

1. $\mathscr{B}$ knows d_w (from $g_w = (g^a)^{d_w}$). Then, $\mathscr{B}$ computes the value of the exponent t such that the value g_w^{bt} deletes the multiple of g^{ab} that comes in the other terms of the signing element $S_{i,j}^{(1)}$. It finally sets $S_{i,j}^{(2)} = (g^b)^t$.
2. $\mathscr{B}$ does not know d_w (from $g_w = (g^a)^{d_w}$). However, it can sometimes fix the exponent $a_j = H_4(m_j, I_j^{(D)}, w)$ to be the unique value in Z_p^* such that the multiples of g^{ab} cancel out in the signing element $S_{i,j}^{(1)}$. Hence, $\mathscr{B}$ is able to generate a valid signature. If the unique value a_j is divulged for identity $I_j^{(D)}$, then $\mathscr{B}$ is allowed to re-use this trick later.

Suppose now that $\mathscr{B}$ does not abort, $\mathscr{A}$ gives a forgery on identity $I_j^{(D)}$, message m_j and dummy message w for which the exponents $\mu_{j,0}$ and $\mu_{j,1}$ (from $g_{j,0}^{\beta} = g^{b\mu_{j,0}}$ and $g_{j,1}^{\beta} = g^{b\mu_{j,1}}$) are not known, and the exponent a_j is not determined regarding the aforementioned trick. Then, $\mathscr{B}$ obtains the value of g^{ab} with high probability given the forgery of the adversary.

6 Application to IoT Networks

We illustrate the practicality of our 2-IBAS scheme ensuring secure bootstrap and aggregation response validation in IoT networks with an example of application.

Intelligent buildings have been designed to accurately control humidity, ventilation, air conditioning, etc, in order to reduce energy waste. When constructing such buildings, sensors are installed in multiple locations and measure room occupancy, temperature, air flow and other parameters. By regularly monitoring physical and technical parameters of intelligent buildings, sudden events such as earthquakes can rapidly and effectively be noticed. In addition, sensors help to supervise mechanical stress at post-event stages.

Let a network of sensors be embedded into an intelligent building. The party representing the network is a device that is accessible by maintenance workers checking the current condition of the building. Sensors are disseminated everywhere in the building and have distinct functionalities. In order to differentiate the functionalities of sensors, we proceed as follows: for instance, let two sensors be designed for levying the humidity, thus sharing the same basic role. However, one sensor is located in room 1 while the second sensor is situated in room 2. Hence, functionalities of these two sensors vary in where humidity is monitored. By doing so, we ensure that no two sensors share the same role, and thus enable locally unique identification.

On regular intervals, the device on behalf of the IoT network is initiated by a maintenance worker. The device sends a request to all the sensors connected to it. The request asks the sensors about their current condition (are the collected data on a good level? are they working well?), and the sensors can reply to it with a personal message such as "humidity level in room 1: ok; condition: ok". A first sensor starts the replying process by generating its personal answer and signing it. It finally forwards the resulting signature to a subsequent sensor. The latter generates it personal answer and signs it using the element given by the first sensor. It forwards its signature to a third sensor. The signing process is

repeated until reaching the last sensor located in the network of the building. Once the signature of the last sensor is computed, the latter gives it to the network's device. The maintenance worker collects the aggregate signature and verifies the current condition of the sensors and thus of the intelligent building.

7 Analysis of the Performance

We implement the 2-IBMS [14] and 2-IBAS schemes by realizing an IoT platform. We present the practicality of the schemes in an IoT environment while considering the limited capacities of such framework, in terms of computation, communication and storage. We examine timing and memory consumption at generation and verification of the signatures.

7.1 Description of the Implementation

The implementation uses the Python-based Charm Crypto library [1], an open source framework developed for rapid prototyping of cryptographic systems. The implementation consists of three Raspberry Pi Model 3 Type B computers, with 64MB of RAM, and is set as follows:

- The first computer is designed as the trusted third party, network and verifier: it distributes the key material (for the network and devices) and verifies the validity of aggregate signature.
- The second computer represents the Device 1: it signs either the message which has been given by the network or by the Device 2 (2-IBMS), or its personal message (2-IBAS). It then forwards the signature to the Device 2.
- The last computer is the Device 2: it signs the message which has been forwarded by the Device 1 (2-IBMS), or its personal message (2-IBAS). It then either sends the signature to the Device 1, such that the ping-pong process is repeated until reaching the maximum number of simulated devices, or forwards the signature to the verifier.

We set a TLS connection between the network and the devices, permitting to forward the key material to the devices in a secure way. We assume that the network is authenticated using a certificate issued by a trusted third party. All local communication modes are supported for maximum flexibility. The devices are able to connect via a LAN interface. The Raspberry Pi Model 3 Type B computers have an integrated WiFi adapter; hence, the devices are configured to use the WiFi connection, providing a real scenario in the IoT context. Moreover, the devices are able to build a wireless ad-hoc network in order to let the communication be independent from any other base station.

7.2 Challenges and Settings

In the 2-IBMS scheme [14], the devices should sign a common message; however, nothing is specified about the nature of this message. Hence, we suggest three cases: the devices sign either an explicit message (as a small sentence), or a "true"/"false" message (as the answer to a closed question), or an empty message (as a positive answer to a closed question). While the order followed by the devices in the signing process is not important, it turns to be difficult to discover which devices did not sign the common message. Hence, a list of devices who did (not) sign could be embedded in the signature to help the verifier. We examine the three settings and show that they remain realistic and usable regarding the IoT environment requirements.

In the 2-IBAS scheme, the devices can personalize the messages to be signed. In order to accurately check the validity of the aggregate signature, the verifier must know the order used for the signing process, such that messages and identities of devices have to be exactly placed in this order. A solution is to

embed messages in the correct order along with the signature; however, this incurs additional communication costs. We study the trade-off between efficiency of the protocol, size of messages and number of devices.

7.3 Basic Timing Analysis

The two schemes are evaluated on the Raspberry Pi computers by running the algorithms for generating the public parameters, the keys and the signature, and for verifying the latter. The Python script takes as argument the number of devices to be simulated and performs the aforementioned phases. The procedure is repeated one thousand times and takes the mean of all the measurements for each assigned number of devices, and is profiled using the Python-based cProfile library[1]. Such measuring method enables to benchmark the precise time consumption while excluding additional overhead from messages being transferred over the network. In the following graphs, we consider 8Bytes-length messages.

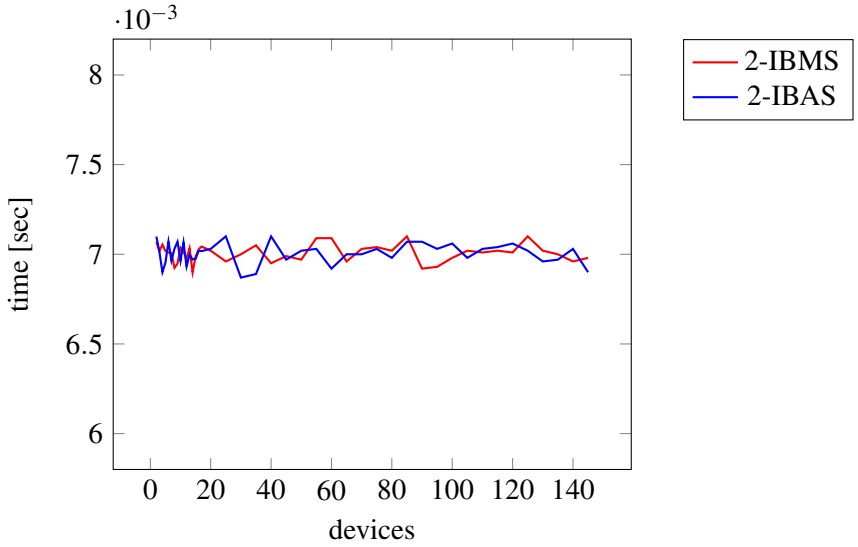


Figure 2: Average running times of Setup and KeyGen_1 regarding the number of devices within the network.

The timing consumption for generating the master secret key of the trusted third party and the key of the network is depicted in Figure 2. The number of devices in the network does not affect the time of this phase. Moreover, it is executed only once. For both schemes, it needs approximately 7ms to generate the key material.

[1] https://docs.python.org/3.5/library/profile.html

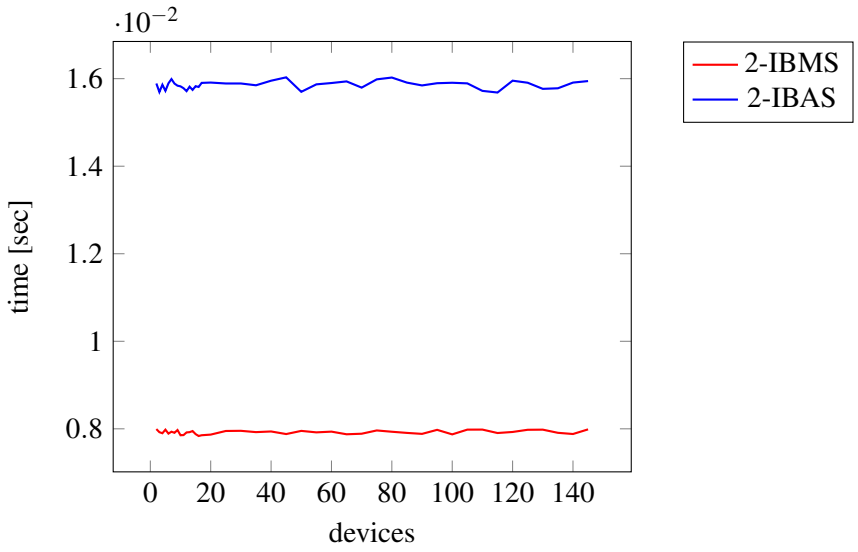


Figure 3: Average running times of KeyGen_2 regarding the number of devices within the network.

The generation of the key for each device requires a constant time (see Figure 3). In the 2-IBMS scheme, one key component is created in 8ms, while in the 2-IBAS scheme, two components are produced in rather 16ms.

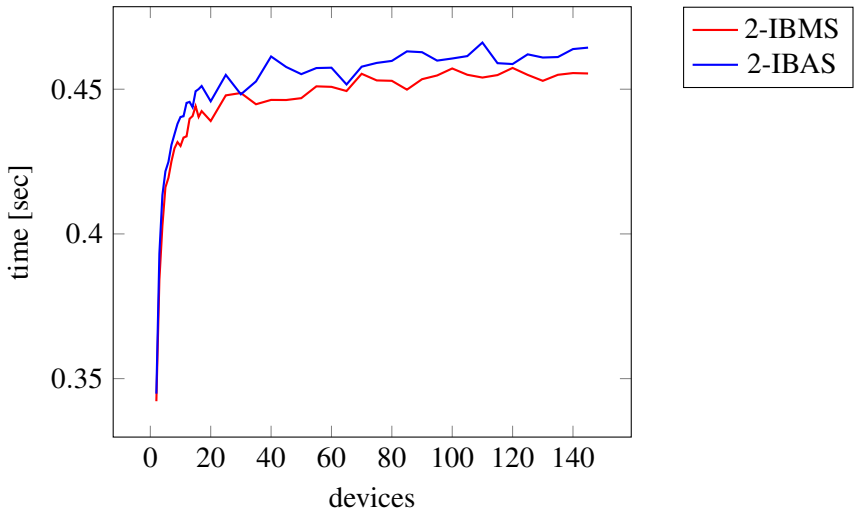


Figure 4: Average running times of Sign_j regarding the number of devices within the network.

The timing consumption for generating the signature seems to depend on the number of devices in the network, as illustrated in Figure 4. Measurements appear to follow a logarithmic development of the time.

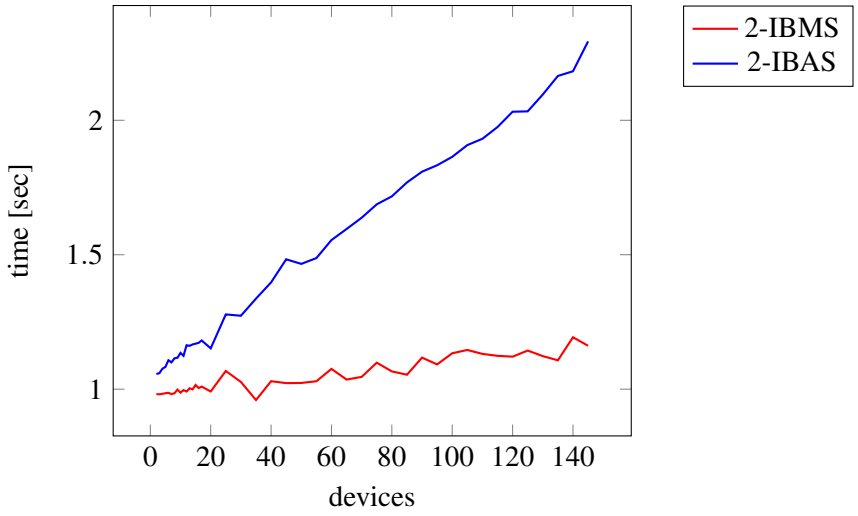


Figure 5: Average running times of Verif regarding the number of devices within the network.

Time for verification of the signature increases linearly with the number of devices in the network (see Figure 5). The gradient of the 2-IBAS scheme ($8.61e-3$) is larger than the gradient of the 2-IBMS scheme ($2.06e-3$).

7.4 Timing Analysis for 2-IBMS

We simulate the 2-IBMS protocol by varying the number of devices in the network such that signatures are generated and exchanged among the two Raspberry Pi computers until the total number of devices is reached. Then, the last signature is forwarded to the verifier (i.e. the third Raspberry Pi computer), who checks its validity. Calculating execution time is done on network transitions, signing and verifying, and the average of one thousand trials is stored. As mentioned in Section 7.2, three settings are made possible for the implementation of the 2-IBMS scheme:

1. Explicit messages contain one or several words and cover all the dictionary. A valid signature means that all the devices have signed the same explicit message. We compare the execution time in function of the length of the message (see Figure 6).
2. There are two (binary) messages in response to a closed question: one for valid state set as "true" and one for invalid state set as "false". A valid signature means that all the devices have signed the "true" message. We compare the execution time in function of the probability p of failure (see Figure 7).
3. The empty message corresponds to the device's answer "true" to a closed question. A valid signature means that all the devices have signed the empty message. We compare the execution time in function of the probability p of failure (see Figure 8).

While the 2-IBMS protocol can prove that all the devices identify themselves at a valid state, it turns to be difficult to find which of the device(s) has(ve) not signed the message in case of failure. In function of the three aforementioned cases, we say that a device fails when it either signs a different message compared to the one signed by the majority of devices, or signs the message "false", or does not sign at all, respectively. The probability p of failure and the number of devices within the network influence the speed of brute-forcing the list of signing devices. It appears to be feasible to brute-force the list of devices for a small probability p (say, one or two devices have signed differently) but to be computationally

expensive for a higher (more realistic) probability p. Hence, instead of letting the verifier brute-forces the list of devices to find which of them failed, we enable the devices to communicate to the verifier when they are at an invalid state. Providing the list of devices who signed not as the majority allows a faster verification. For instance, we can encode each device with one byte and add a maximum of 256Bytes to the network payload (assuming that there are less than 256 devices in the network), which does not influence the speed of network transmissions.

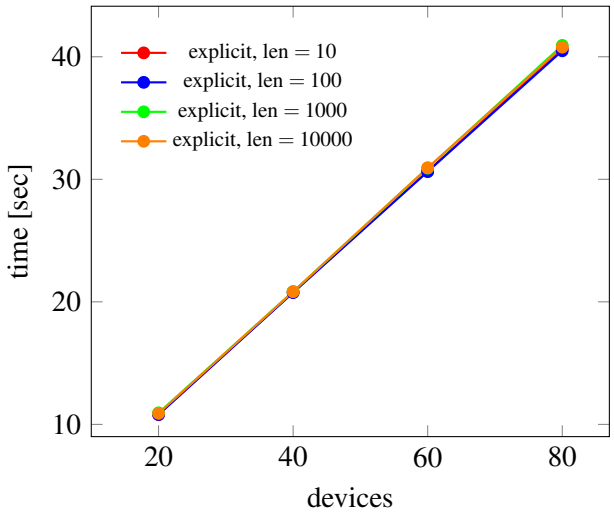


Figure 6: Average running times of signature generation and verification on explicit messages for different lengths and regarding the number of devices within the network.

Figure 6 depicts the case where devices sign explicit messages. We test this implementation with different lengths of messages. The length of the message should influence the speed of the round trip, that includes the generation of the signature, transfer of packets over the network and verification of the signature. A slightly longer time to send the packets over the WIFI network is expected. With a small increase of the message length, the execution time remains almost constant.

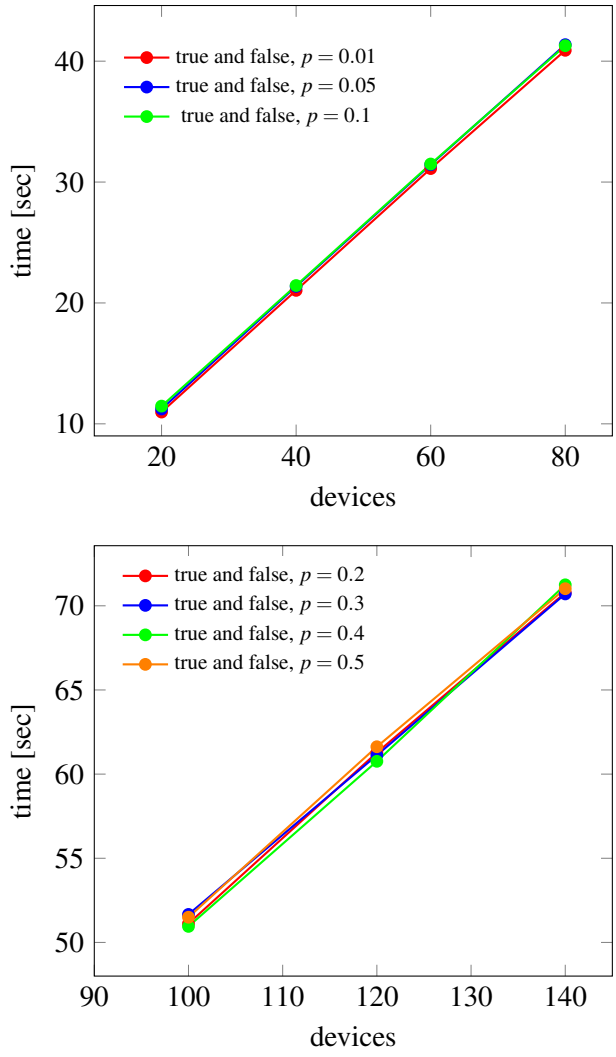


Figure 7: Average running times of signature generation and verification on “true” and “false” messages for different values of the probability p of failure and regarding the number of devices within the network.

Figure 7 represents the case where devices sign a “true”/”false” message in function of their state, either valid or invalid. A list of devices who signed the message “true” can be provided to the verifier in order to optimize the performance. The execution time linearly increases with the number of devices within the network. Moreover, the time required for generating and verifying signatures is independent from the failure probability p. For a high probability p and a big number of devices, there is a small increase of the round-time due to larger packets which are transferred over the WIFI network.

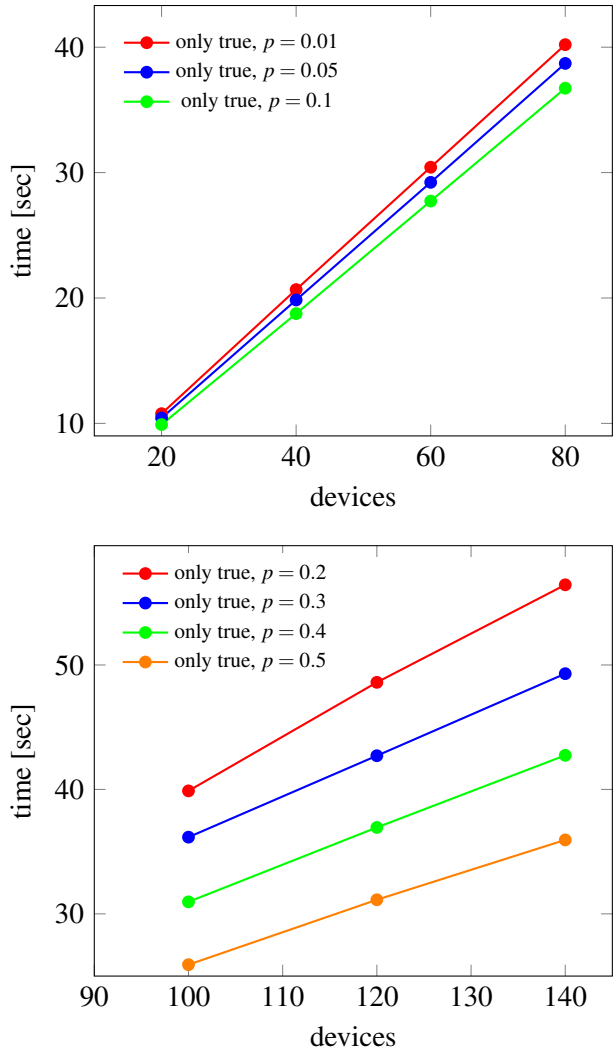


Figure 8: Average running times of signature generation and verification on empty messages ("true") for different values of the probability p and regarding the number of devices within the network.

Figure 8 describes the case where devices only sign an empty message when their state is valid, and do not sign otherwise. A list of devices who signed can be provided to the verifier in order to optimize the performance. The execution time linearly increases with the number of devices within the network. Moreover, higher the probability of failure p is, faster the signature generation and verification are.

Figure 9 combines the results on empty ("true") and "true"/"false" messages. Timing gaps are more noticeable for higher numbers of devices within the network. Signing an empty message as an answer for valid state appears to be the optimal solution, subject to providing the list of (non-)signing devices. Workload is favorable on devices' and verifier's sides if all the devices have signed the empty message.

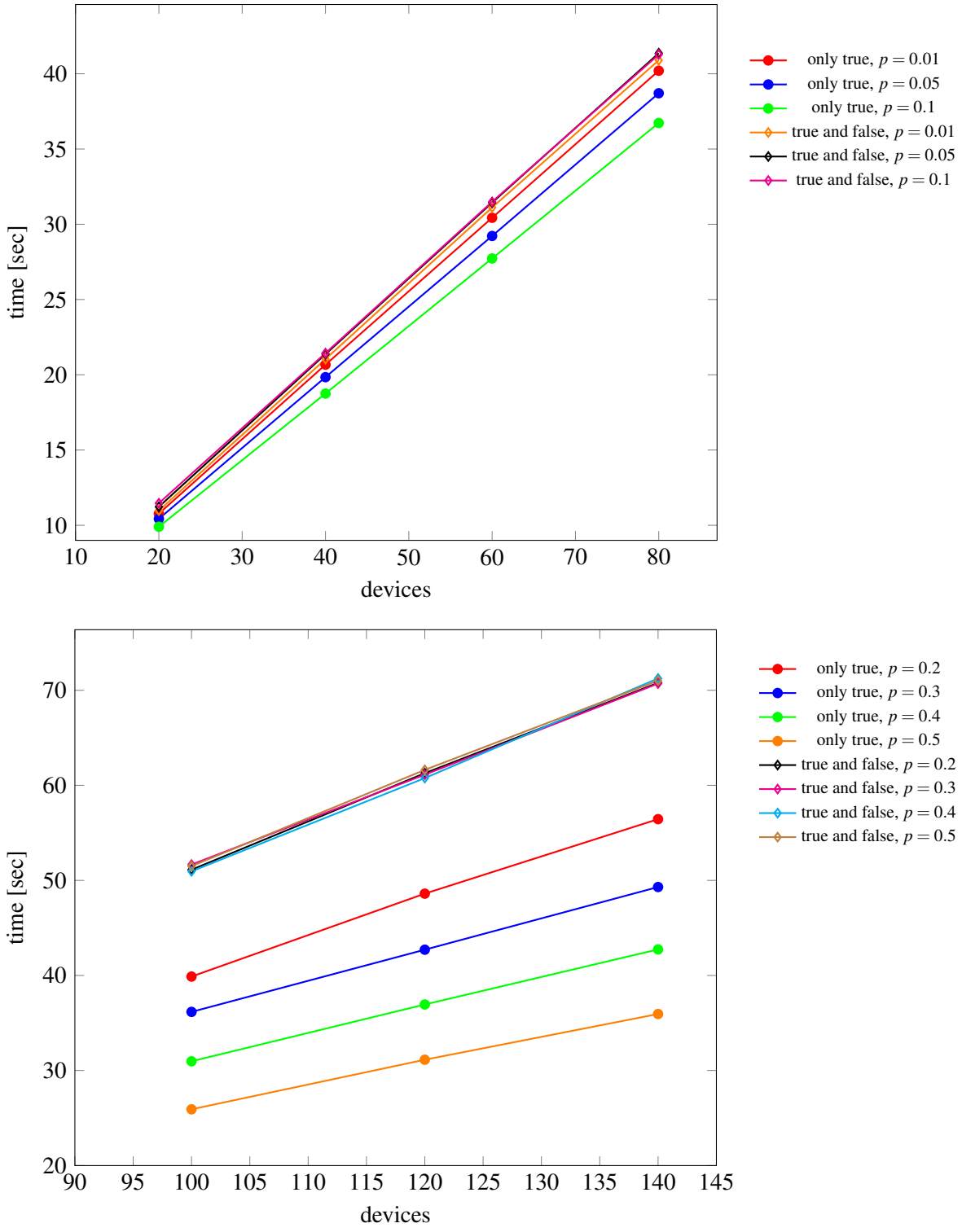


Figure 9: Average running times of signature generation and verification on empty (“true”) and “true”/”false” messages and regarding the number of devices within the network.

7.5 Timing and Memory Analysis for 2-IBAS

The 2-IBAS scheme enforces an order during the signing process. This order includes devices' identities and their chosen messages, and is required for verification. An option to optimize the execution of the protocol is to forward a sequence of identities representing the fixed order to the verifier.

Experiments are done among two Raspberry Pi computers representing the devices signing one after the other, until reaching the assigned number of devices in the network. The last signature is then transferred to the verifier (i.e. the third Raspberry Pi computer), who verifies its validity. Workflow timing includes network transitions, signature generation and verification, and is depicted in Figure 10.

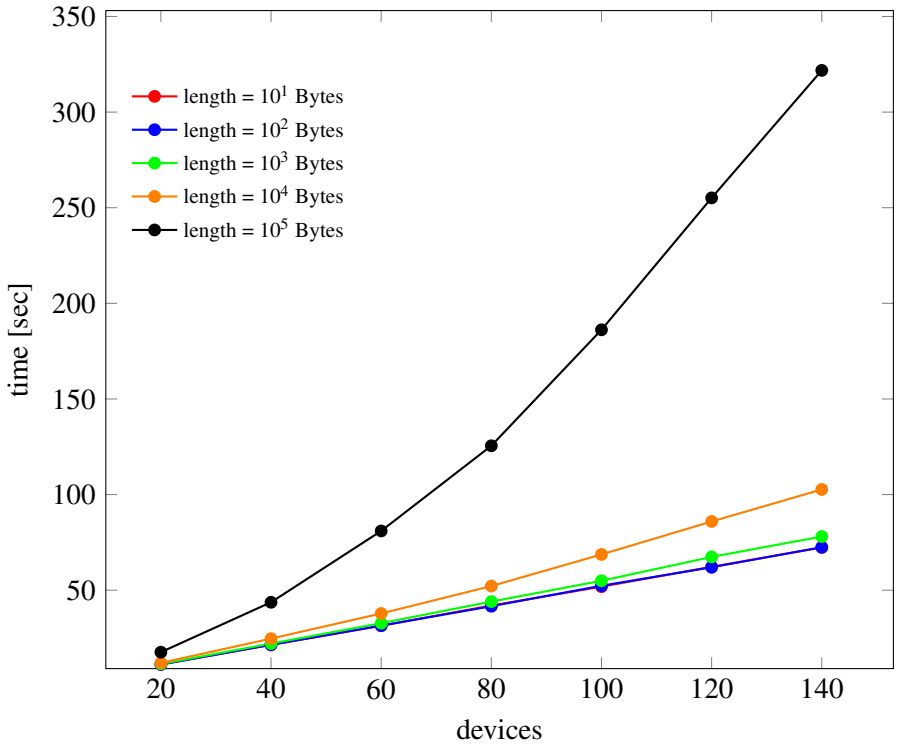


Figure 10: Average running times of signature generation and verification on explicit messages for different lengths and regarding the number of devices within the network.

For reasonable-length messages (say, less than 10^5Bytes), the execution time is linear with the number of devices in the network. However, big-length messages (say, more than 10^5Bytes) impact on the practicality of the protocol. Since the number of devices and the length of messages vary, we choose to calculate the memory consumption at devices' and verifier's sides. We implement the 2-IBAS scheme on a regular laptop (instead on Raspberry Pi computers). Since the architecture remains the same, there should not be any noticeable difference. In order to profile the memory consumed over the time, we use the Python-based memory profiler[2]. We execute one hundred times the script `mprof run` and save the average. Results are illustrated in Figures 11, 12, 13 and 14.

[2]https://github.com/pythonprofilers/memory_profiler

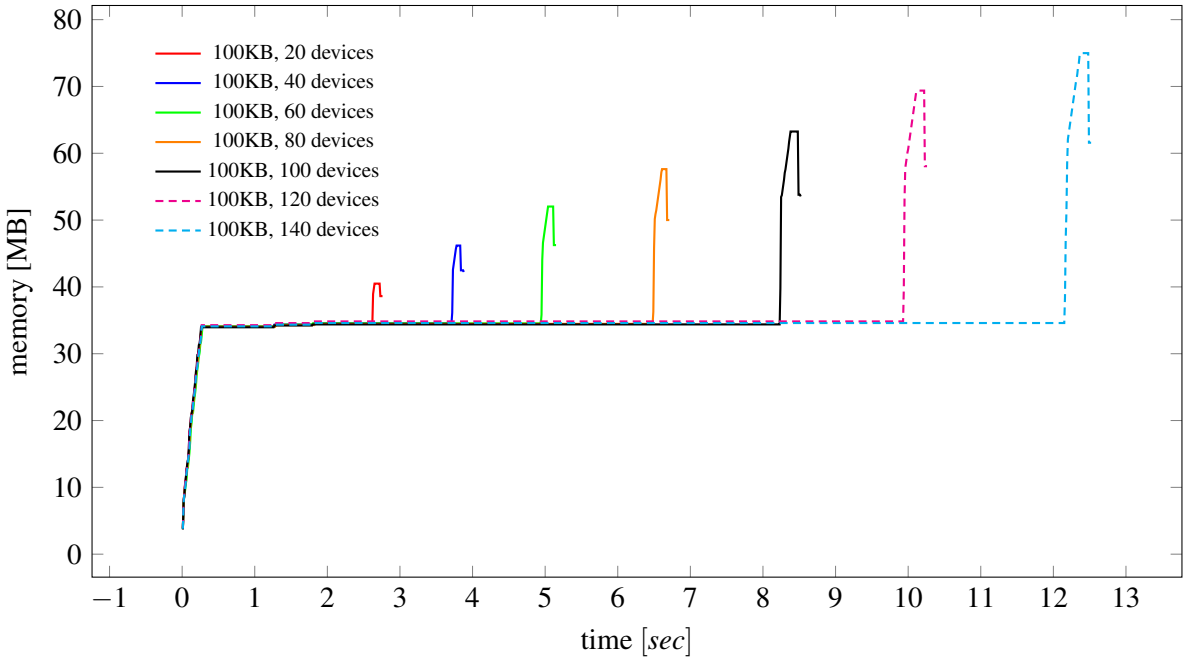


Figure 11: Average memory amount at verifier's side for fixed-length messages and different numbers of devices within the network.

Figure 11 represents the amount of memory allocated by the verifier from the execution of the script at the end of the signature verification, in function of the number of devices in the network. The amount of memory consumed at idle time floats around 37MB, regardless the number of devices that have registered through the network. The required memory increases at the verification stage. Although an increase of roughly the amount necessary to store the signed messages is expected, the actual increase is around three times more (for example, while an additional memory of $140 \times 0.1 = 14$MB is expected for 140 devices and a message of 100KB, the actual additional memory is of 41 MB).

Figure 12 depicts the results at the verifier's side with increasing length of personal messages. Longer the message is, higher the required memory for the verifier is.

Figure 13 illustrates the amount of memory allocated by the device from the execution of the script at the end of the signature generation, in function of the number of devices in the network. Each device receives a message multiple times in order to simulate a network with a high number of connected devices. As the Raspberry Pi computer simulates each time a different device in the sequence of signature generation, the memory consumption increases. Since every device adds its own personal message, devices located at the end of the sequence receive a higher amount of messages, and thus require more memory and space. Such effect is observed in the plots and shows that the memory requisites vary along the sequence. However, the plots are not smooth, increasing curves, but rather present several spikes. Such results might be spotted since the memory for the next simulated device is allocated before the message of the previous simulated device has been removed from it. The amount of allocated memory depends on the number of devices, and devices at the end of the sequence are notably affected by such variation. The amount of memory required in idle state is the same than the one measured at the verifier's side. It represents the consumed memory from the shared scripts among the verifier and the device, the Charm Crypto library and the Python language.

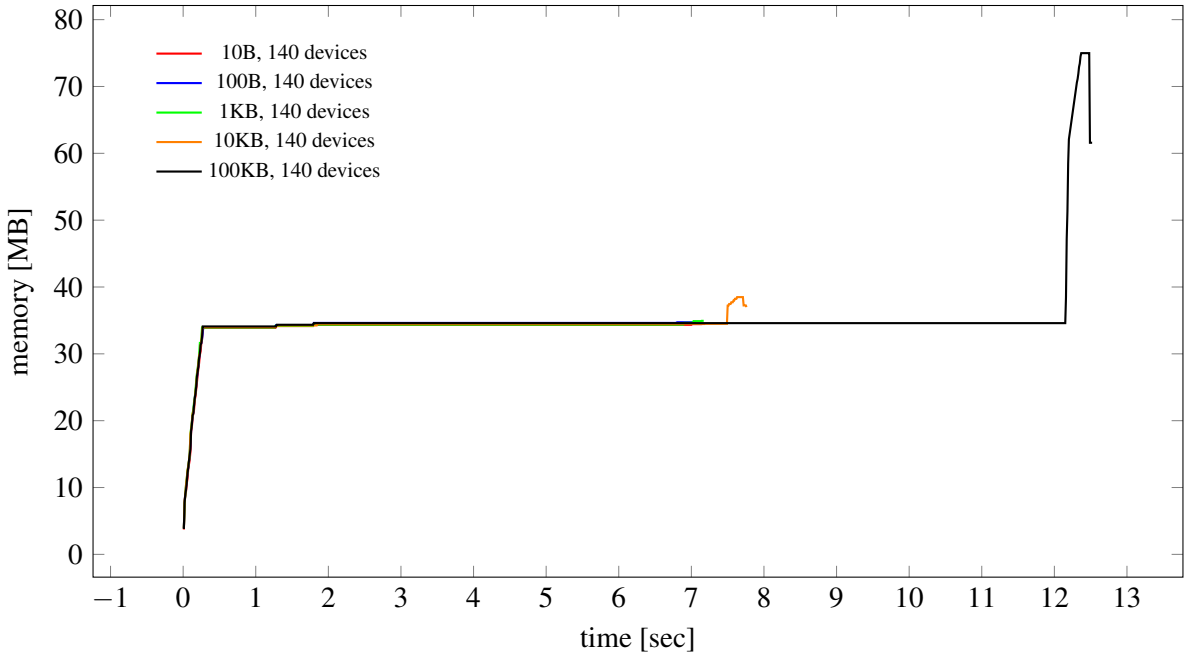


Figure 12: Average memory amount at verifier's side for different lengths of messages and a fixed number of devices within the network.

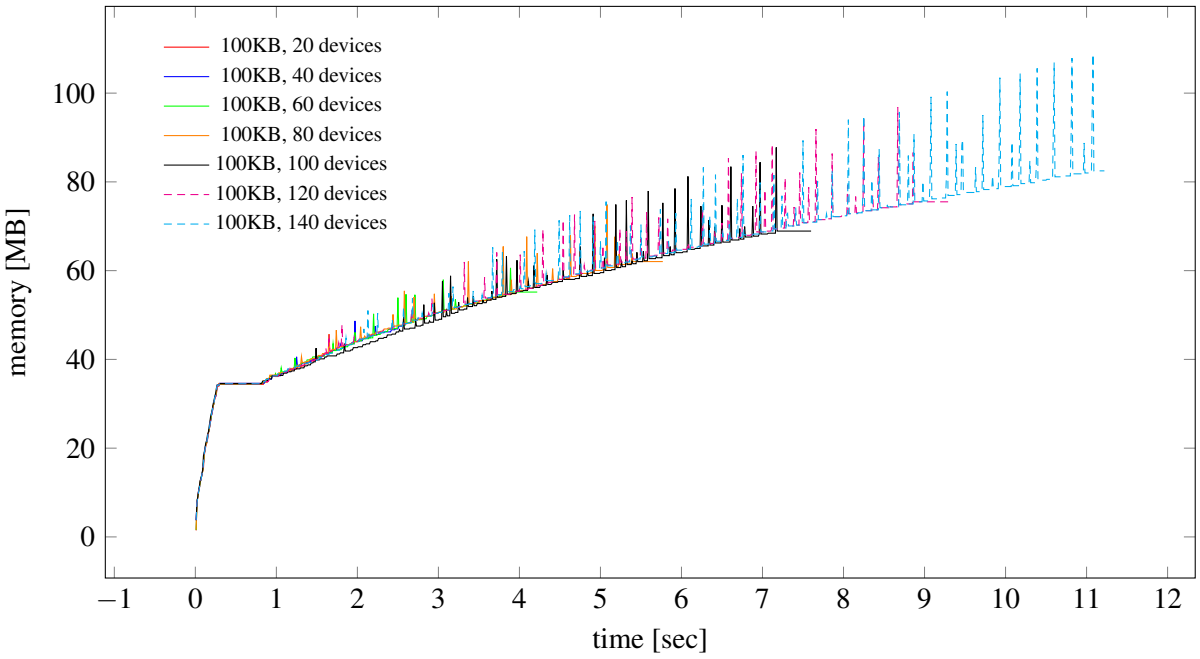


Figure 13: Average memory amount at device's side for fixed-length messages and different numbers of devices within the network.

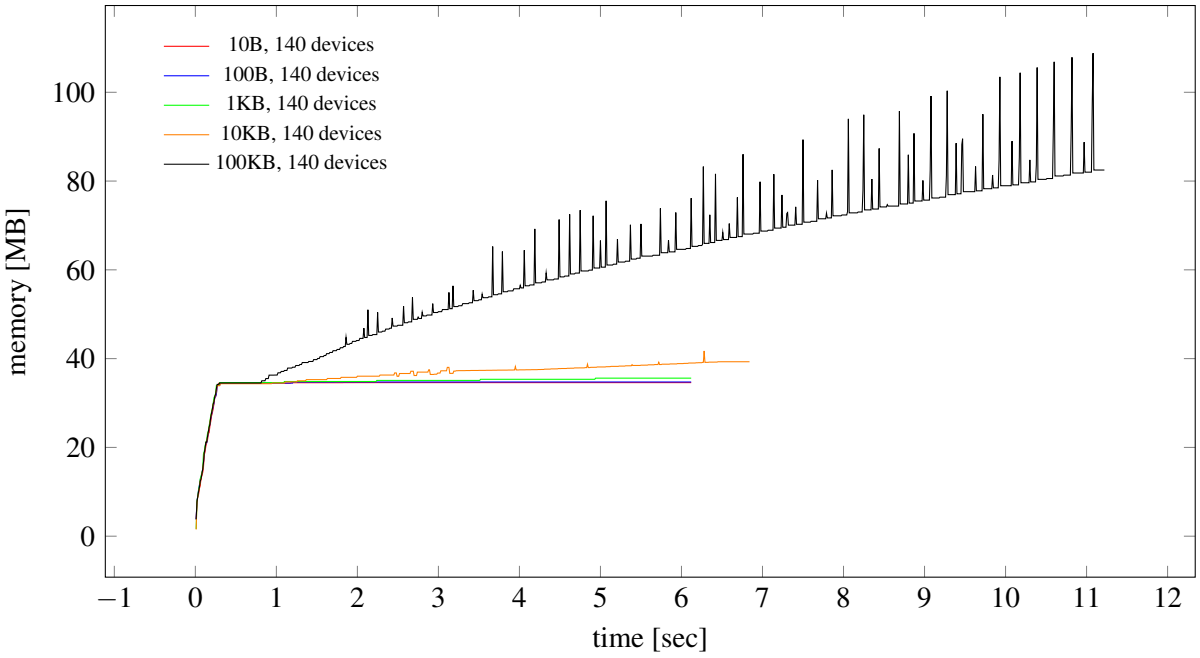


Figure 14: Average memory amount at device's side for different lengths of messages and a fixed number of devices within the network.

Figure 14 depicts the results at the device's side with increasing length of personal messages. The amount of allocated memory increases rapidly with the length of personal messages, and each device in the network is affected by such variation.

8 Related Work

Several works focus on setting identity-based cryptography from PKIs and vice-versa. Implication from an Identity-Based Encryption scheme to a signature scheme is shown in [6]. Transformations from a signature scheme into an IBS scheme using certification techniques are presented in [10, 2]. Extension of Aggregate Signature (AS) into Identity-Based AS (IBAS) is suggested in [11]. However, the size of the aggregate signature depends on the number of signers since the certificates must be included in each signature. Aggregation of signatures in certificates is attempted in [7]. Nevertheless, aggregation of the public keys embedded into the signatures is not possible. Hence, no simple transformation of a compact signature scheme into an identity-based one does exist [3].

Boneh et al. [7] propose an AS scheme that supports flexible aggregation such that anyone can aggregate multiple individual signatures in any order into an aggregate signature. Lysyanskaya et al. [18] require certified trapdoor permutations to support sequential aggregation, that constrain a signer to aggregate its individual signature into the aggregate signature formed by the previous signatures. The two aforementioned schemes allow compact aggregate signatures (i.e. the size does not depend on the number of signers). However, such schemes are not extensible to identity-based settings. An AS scheme [8, 5] can be seen as a variation of a Multi-Signature (MS) scheme [15, 19, 4, 17] where individual to-be-signed messages are selected in the former while a common message is chosen in the latter.

The Identity-Based MS (IBMS) scheme in [3] enables to generate the signing keys according to identities of parties and to compute a multi-signature with participation of all these parties. However, the set of identities of the signers must be known beforehand since the signature generation algorithm

is run with this set as input. In addition, the signed message must be common to all signers and there is no hierarchy of parties. Gentry and Ramzan [12] give IBMS and IBAS schemes such that the IBMS scheme serves as a base for the IBAS one. In the latter, the signers create their own signatures on distinct messages, and then aggregate them at the end to obtain the aggregate signature. This can be done sequentially, taking as input the signature of the previous signer to generate the signature of the current signer. Again, no hierarchy can be supported.

A Hierarchical Identity-Based Signature (HIBS) scheme is defined in [13] where the number of levels in the hierarchy is arbitrary. However, no signature aggregation is possible. A multi-key HIBS scheme is presented in [16] where signers are placed following a hierarchical path and generate a non-compact aggregate signature. Nevertheless, in our case, two levels are enough since only the network should be at the top while the devices should be all siblings at the first level. Gritti et al. [14] present a 2-level IBMS, suitable for local IoT-based applications. Secret and signing components have constant size and costly operations are executed during the verification phase. However, their scheme is limited to the case of IBMS, which constraints the IoT devices to sign pre-chosen common messages instead of their own personal ones.

9 Conclusion

Reliable identity and data management in IoT networks has been challenging due to the diversity of devices' provenance and functionality. In addition, with the restricted computing, storing and transmitting resources of devices, identity and data authentication should be done efficiently. We proposed a provably secure 2-level Identity-Based Aggregate Signature scheme that enables secure bootstrap and message attestation. Thanks to locally unique identification of devices along with aggregation of devices' responses for validation, our solution is applicable in IoT networks.

Acknowledgments

This work was supported by the EU FP7 ERANET program under grant CHIST-ERA-2016 UPRISE-IOT.

References

[1] J. A. Akinyele, M. D. Green, and A. D. Rubin. Charm: A framework for rapidly prototyping cryptosystems. Cryptology ePrint Archive, Report 2011/617, 2011.

[2] M. Bellare, C. Namprempre, and G. Neven. Security proofs for identity-based identification and signature schemes. In *Proc. of the 23rd International Conference on the Theory and Applications of Cryptographic Techniques (EUROCRYPT'04), Interlaken, Switzerland*, Lecture Notes in Computer Science, pages 268–286. Springer-Verlag, May 2004.

[3] M. Bellare and G. Neven. Identity-based multi-signatures from rsa. In *Proc. of the 7th Cryptographers' Track at the RSA Conference on Topics in Cryptology (CT-RSA'07), San Fancisco, CA, USA*, Lecture Notes in Computer Science, pages 145–162. Springer-Verlag, February 2006.

[4] A. Boldyreva. Threshold signatures, multisignatures and blind signatures based on the gap-diffie-hellman-group signature scheme. In *Proc. of the 6th International Workshop on Theory and Practice in Public Key Cryptography: Public Key Cryptography (PKC'03), Miami, USA*, Lecture Notes in Computer Science, pages 31–46. Springer-Verlag, January 2003.

[5] D. Boneh. *Aggregate Signatures*, pages 27–27. Springer-Verlag, 2011.

[6] D. Boneh and M. Franklin. Identity-based encryption from the weil pairing. In *Proc. of the 21th Annual International Conference on Advances in Cryptology (CRYPTO'01), Santa Barbara, California, USA*, Lecture Notes in Computer Science, pages 213–229. Springer-Verlag, August 2001.

[7] D. Boneh, C. Gentry, B. Lynn, and H. Shacham. Aggregate and verifiably encrypted signatures from bilinear maps. In *Proc. of the 22nd International Conference on Theory and Applications of Cryptographic Techniques (EUROCRYPT'03), Warsaw, Poland*, Lecture Notes in Computer Science, pages 416–432. Springer-Verlag, May 2003.

[8] D. Boneh, C. Gentry, B. Lynn, and H. Shacham. A survey of two signature aggregation techniques. *CryptoBytes*, 6(2), 2003.

[9] K. Butler, T. R. Farley, P. McDaniel, and J. Rexford. A survey of bgp security issues and solutions. In *Proc. of the IEEE*, volume 98, pages 100–122. IEEE, Jan 2010.

[10] Y. Dodis, J. Katz, S. Xu, and M. Yung. Strong key-insulated signature schemes. In *Proc. of the International Conference on Public Key Cryptography (PKC'03), Miami, USA*, Lecture Notes in Computer Science, pages 130–144. Springer-Verlag, January 2003.

[11] D. Galindo, J. Herranz, and E. Kiltz. On the generic construction of identity-based signatures with additional properties. In *Proc. of the 12th International Conference on Theory and Application of Cryptology and Information Security (ASIACRYPT'06), Shanghai, China*, lecture Notes in Computer Science, pages 178–193. Springer-Verlag, December 2006.

[12] C. Gentry and Z. Ramzan. Identity-based aggregate signatures. In *Proc. of the 9th International Conference on Theory and Practice of Public-Key Cryptography (PKC'06), New York, NY, USA*, Lecture Notes in Computer Science, pages 257–273. Springer-Verlag, April 2006.

[13] C. Gentry and A. Silverberg. Hierarchical id-based cryptography. In *Proc. of the Advances in Crypotology 9th International Conference on Theory and Application of Cryptology and Information Security (ASIACRYPT'02), Queenstown, New Zealand*, Lecture Notes in Computer Science, pages 548–566. Springer-Verlag, December 2002.

[14] C. Gritti, R. Molva, and M. Önen. Lightweight secure bootstrap and message attestation in the Internet of Things. In *Proc. of the 33rd ACM Symposium On Applied Computing (SAC'18), Pau, France*, pages 775–782. ACM, April 2018.

[15] K. Itakura and K. Nakamura. A public key cryptosystem suitable for digital multi-signatures. *NEC Research and Development*, 71:1–8, 1983.

[16] H. W. Lim and K. G. Paterson. Multi-key hierarchical identity-based signatures. In *Proc. of the 11th IMA International Conference in Cryptography and Coding (Cryptography and Coding'07), Cirencester, United Kingdom*, Lecture Notes in Computer Science, pages 384–402. Springer-Verlag, December 2007.

[17] S. Lu, R. Ostrovsky, A. Sahai, H. Shacham, and B. Waters. Sequential aggregate signatures and multisignatures without random oracles. In S. Vaudenay, editor, *Proc. of the 24th Annual International Conference on the Theory and Applications of Cryptographic Techniques (EUROCRYPT'06), Saint Petersburg, Russia*, Lecture Notes in Computer Science, pages 465–485. Springer-Verlag, May 2006.

[18] A. Lysyanskaya, S. Micali, L. Reyzin, and H. Shacham. Sequential aggregate signatures from trapdoor permutations. In *Proc. of the International Conference on the Theory and Applications of Cryptographic Techniques (EUROCRYPT'04), Interlaken, Switzerland,*, Lecture Notes in Computer Science, pages 74–90. Springer-Verlag, May 2004.

[19] S. Micali, K. Ohta, and L. Reyzin. Accountable-subgroup multisignatures: Extended abstract. In *Proc. of the 8th ACM Conference on Computer and Communications Security (CCS'01), Philadelphia, PA, USA*, pages 245–254. ACM, November 2001.

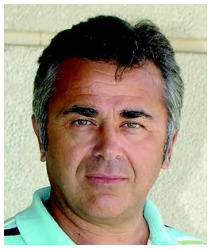
[20] K. Seo, C. Lynn, and S. Kent. Public-key infrastructure for the secure border gateway protocol (s-bgp). In *Proc. of the DARPA Information Survivability Conference (DISCEX'01), Anaheim, CA, USA*, pages 239–253. IEEE, June 2001.

[21] A. Shamir. Identity-based cryptosystems and signature schemes. In *Proc. of the Annual International Cryptology Conference on Advances in Cryptology (CRYPTO'84), Santa Barbara, CA, USA*, Lecture Notes in Computer Science, pages 47–53. Springer-Verlag, August 1985.

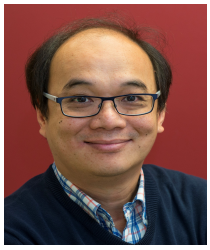
Author Biography

Clémentine Gritti received a Ph.D. degree in Computer Science from University of Wollogong in 2017. She was a post doctoral researcher within Eurecom, France in 2017-2018, working on the Internet of Things. She recently joined the NaCl group in NTNU, Norway as a post doctoral researcher. Her current project focuses on electronic voting systems. Her main research interests include public-key cryptography and information security.

Melek Önen is an Associate Professor in the Digital Security Department at EURECOM (Sophia-Antipolis, France). Her research interests are applied cryptography, information security and privacy. She has worked in the design and the development of cryptographic protocols for various technologies including the cloud and the IoT. Prof. Melek Önen was/is involved in many European projects. She was the scientific leader of the EU H2020 TREDISEC project which was focusing on cloud security. She currently is the coordinator of the EU H2020 PAPAYA project on privacy preserving data analytics. Prof. Melek Önen holds a PhD in Computer Science from Ecole Nationale Supérieure des Télécommunications de Paris (ENST, 2005).



Refik Molva is a full professor and the head of the Digital Security Department at EURECOM in Sophia Antipolis, France. His current research interests are the design and evaluation of protocols for security and privacy in cloud computing. He previously worked on several research projects dealing with security and privacy in social networks, RFID systems, self-organizing systems, and mobile networks. He was program chair or general chair for security conferences such as ESORICS, RAID, SecureComm, IEEE ICC and various security related workshops. He has been an area editor for various journals such as Computer Networks, Pervasive and Mobile Computing, Computer Communications, and the International Journal of Information Security. Beside security, he worked on distributed multimedia applications over high speed networks and on network interconnection. Prior to joining Eurécom, he worked in the Zurich Research Laboratory of IBM where he was one of the key designers of the KryptoKnight security system. He also worked as a consultant in security for the IBM Consulting Group. Refik Molva has a Ph.D. in Computer Science from the Paul Sabatier University in Toulouse (1986) and a B.Sc. in Computer Science (1981) from Joseph Fourier University, Grenoble, France.



Willy Susilo received a Ph.D. degree in Computer Science from University of Wollongong, Australia. He is a Professor and the Head of School of Computing and Information Technology and the director of Institute of Cybersecurity and Cryptology (iC2) at the University of Wollongong. He was previously awarded the prestigious ARC Future Fellow by the Australian Research Council (ARC) and the Researcher of the Year award in 2016 by the University of Wollongong. His main research interests include cybersecurity, cryptography and information security. His work has been cited more than 9,000 times in Google Scholar. He is the Editor-in-Chief of the Information journal. He has served as a program committee member in dozens of international conferences. He is currently serving as an Associate Editors in several international journals, including Elsevier Computer Standards and Interface and International Journal of Information Security (IJIS, Springer). He has published more than 400 research papers in the area of cybersecurity and cryptology.

Thomas Plantard acquired his PhD on computer science at the University of Montpellier II in 2005 on the subject of computer arithmetic for cryptography. Dr. Thomas Plantard is currently a senior research fellow at the Institute of Cybersecurity and Crypotology that he have joined in 2006. His research is focused post-quantum cryptography: cryptography resistant even to quantum computer aided attacks. In particular, Dr Thomas Plantard has acquired expertise in lattice based cryptology, on both cryptography and cryptanalysis side.