CHARIOT: Cloud-Assisted Access Control for the Internet of Things

Clémentine Gritti, Melek Önen, Refik Molva
EURECOM
Sophia Antipolis, France
name.surname@eurecom.fr

***Abstract*—The Internet of Things (IoT) technology has expanded widely across the world, promising new data management opportunities for industries, companies and individuals in different sectors, such as health services or transport logistics. The exponentially increasing number of IoT devices, their origin diversity, their limited capabilities in terms of resources, as well as the ever-increasing amount of data, raise new challenges for security and privacy protection, precluding traditional access control solutions to be integrated to this new environment. In this paper, we propose a reliable, server-aided, policy-based access control mechanism, named CHARIOT, that enables an IoT platform to verify credentials of different devices requesting access to the data stored within it. CHARIOT enables IoT devices to authenticate themselves to the platform without compromising their privacy by using attribute-based signatures. Our solution also allows secure delegation of costly computational operations to a cloud server, hence relieving the workload at IoT devices' side.**

***Index Terms*—Access Control, Cloud Computing, Internet of Things**

I. Introduction

The Internet of Things (IoT) has been developed to enable the interconnection between various devices, such as mobile phones, sensors and actuators, that collect and transmit data at large-scale. While many platforms are available for the IoT, access control issues are often overlooked; various attacks against such platforms have been noticed[1]. Preventing unauthorized access to the platform and consequently to the data stored in there, becomes extremely challenging due to the nature of IoT devices. The latter, that are often simple and resource-constrained, cannot perform costly computational operations. Since platforms are usually installed at untrusted third parties, such as data centers or cloud servers, such parties must not obtain any information about the identity of IoT devices and the data that is being collected or accessed.

Traditional credential-based access control systems do not suit this environment as resource-limited IoT devices are often not able to generate credentials or signatures to satisfy the access control policy defined for an IoT platform. We hence aim to develop a server-aided access control solution, named CHARIOT, which ensures the privacy of the devices' identity and of the data towards the platform. CHARIOT relies on the use of attribute-based signatures to ensure the proper authentication of devices, by defining an access control policy for the data stored at the IoT platform. Furthermore, CHARIOT enables an IoT device to delegate most of the access control operations to a more powerful third party such as a cloud server, and only perform minor operations at its side in order to optimize the use of its resources. The outsourced computation of the credentials is based on a simple secret sharing of the signing key between the cloud server and the IoT device. Once the cloud server has executed the main part of the generation of the signature, the IoT device performs very few additional operations to finalize it and sends the resulting signature to the platform. CHARIOT ensures that no information about the devices' attributes in the credentials is leaked to the platform nor to the cloud server.

In the following section, we detail the problem of designing an access control protocol in the IoT environment and highlight the main features of our cloud server-assisted protocol. Section III describes the proposed solution CHARIOT. In Section IV, we evaluate the performance of our solution and recall the existing work. We finally conclude the paper in Section V.

II. Problem Statement

In the IoT context, giving access to sensitive information to unauthorized parties brings serious issues on security [1]. Access control systems appear to be the essential strategy to overcome these threats. Nevertheless, traditional access control solutions relying on public-key infrastructures fall short for this technology mainly because of the very limited computing capabilities of IoT devices and storage.

A suitable technique allowing the secure authentication of devices to the platform is Attribute-Based Signature (ABS) [2]. Informally speaking, an ABS protocol in the IoT environment ensures that given an access policy, whenever a device signs a message using its attributes, if the attributes satisfy this access policy, then this signature is valid and the device successfully authenticates and accesses the platform. Additionally, ABS ensures that these attributes remain hidden in the signature, and thus the device's identity remains private. However, current ABS solutions [3], [4] suffer from the computational burden of the signature generation.

In most of the existing ABS solutions, the generation of the signature and its size depend on the number of attributes in the

[1]https://motherboard.vice.com/en_us/article/4xav93/more-security-vulnerabilities-found-in-hello-barbie-toys-servers (25/01/16)

signing policies. Yet, in the context of an IoT environment, a typical access control policy may contain numerous attributes because of the large number of IoT devices and their heterogeneity. Herranz et al. [4] present a Threshold ABS scheme with compact signatures whereby their size does not depend on the number of attributes in the policies. While the solution is suitable for the IoT technology, the computational cost still remains significant: the number of modular exponentiations is linear with the number of attributes and therefore cannot be afforded by resource-constrained IoT devices.

We propose to improve the computational cost of the above solution at the device's side by delegating most of the signature generation to a powerful cloud server. Similarly to [5], the signing key is secretly shared between the cloud server and the IoT device, ensuring a secure delegation of the signature computation. The cloud server computes a partial signature using an outsourcing key, and sends it to the IoT device. The latter finalizes the signature by performing additional lightweight operations using its private key, and forwards it to the platform. The signature is accepted by the platform if the device's attributes, embedded into the signature, satisfy the access control policy.

Yet, the ABS solution in [4] incurs extra computational and storage overhead since the signature generation is fixed to an upper bound due to the use of dummy attributes. Following a technique introduced in [6], we modify the original ABS in [4] by removing the presence of dummy attributes. We hence obtain a more efficient Threshold ABS scheme with constant-size signatures and no dummy attributes whereby the most computationally-intensive operations are securely delegated to a cloud server. The proposed solution guarantees privacy of participating IoT devices against the platform and cloud server.

III. CHARIOT

A. Preliminaries

Let $\mathbb{G}$ and $\mathbb{G}_T$ be two cyclic multiplicative groups of prime order p and $e : \mathbb{G} \times \mathbb{G} \rightarrow \mathbb{G}_T$ be an admissible pairing with properties of bilinearity, non-degeneracy and computability. Let $g \in \mathbb{G}$ and $\vec{v} = (v_1, v_2, v_3)^\top \in \mathbb{G}^3$. We denote $E(g, \vec{v})$ the pairing-based vector $(e(g, v_1), e(g, v_2), e(g, v_3))^\top \in \mathbb{G}_T^3$. We let the multiplication between two column vectors be $(v_1, v_2, v_3)^\top \cdot (v'_1, v'_2, v'_3)^\top = (v_1 \cdot v'_1, v_2 \cdot v'_2, v_3 \cdot v'_3)^\top \in \mathbb{G}^3$. Let Y be a finite set and $y \in_R Y$ be a random variable uniformly chosen from Y. Let X be an attribute set, $at \in X$, τ be an injective encoding such that all $\tau(at)$ are pairwise-distinct, and $\gamma \in \mathbb{Z}_p$. We denote $F_X(\gamma) = \prod_{at \in X}(\gamma + \tau(at))$ the polynomial of degree $|X|$.

B. Building Blocks

Similarly to Herranz et al. [4], our ABS scheme relies on two main features, namely the Attribute-Based Encryption (ABE) scheme with constant-size ciphertexts proposed by [7] and the Groth-Sahai proof systems for bilinear groups [8].

The ABE scheme in [7] is designed for the threshold case, where users are authorized to decrypt if they have at least t attributes matching the ones from an attribute universe, for some threshold t chosen by the party who encrypts the message. Such scheme relies on expressing a polynomial as the product of irreducible factors of degree 1 with coefficients equal to the attributes from either the user's set or the universe. Fraction of such polynomials can thus be simplified when t attributes among the user's set and the universe match.

Herranz et al. [4] design their Threshold ABS scheme by enabling the signer to implicitly prove that it can decrypt a ciphertext generated as in the ABE scheme [7]. To do so, the signer generates a Groth-Sahai proof [8] in which the message and access policy are binded using a technique from [9]. Informally, by hashing the to-be-signed message using Waters' techniques [10] and embedding it into the Groth-Sahai Common Reference String (CRS), the technique in [9] allows signatures of knowledge.

In addition, we modify the ABS scheme presented in [4] by outsourcing the signature generation to a cloud server. Similarly to Chen et al. [5], we let the cloud server and the IoT device hold shares of a secret element chosen by the trusted attribute authority, and use them to generate the signature such that the combination of the two shares recover the secret.

For more details on our building blocks, we let the reader refer to the full version of the paper [11].

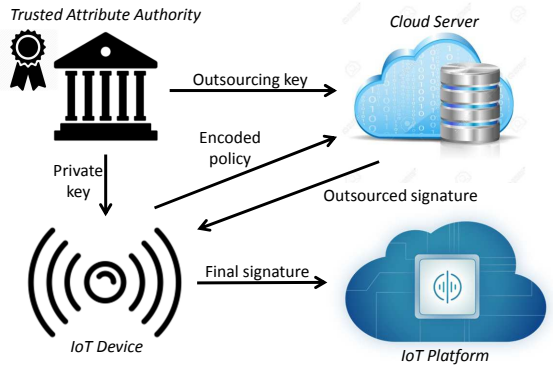
C. Overview

The proposed solutions consists of six algorithms. The Setup algorithm generates the public parameters accessible to all participating parties, and a master secret key forwarded to an off-line trusted attribute authority. The Keygen algorithm, run by the trusted attribute authority, creates the outsourcing key for the cloud server and the private key for the IoT device, regarding the access attributes of this device. The outsourcing key embeds the attributes of the device in their hashed form (using a keyed Hash Message Authentication Code (HMAC)) while the private key is calculated in order to enable the device to complete its access requests to the platform. Informally speaking, the outsourcing key and the device's private key contain shares of a secret enabling to apply the outsourced signature generation technique.

Secure authentication of the IoT device towards the platform is permitted with algorithms Request, Sign_{out}, Sign and Verify. The IoT device runs Request to hash the access policy for the cloud server using the HMAC. The access policy is defined by the platform and made accessible to the device, but should remain hidden from the cloud server's view. Then, the cloud server, given this hashed access policy, creates the outsourced signature using its outsourcing key. It forwards the outsourced signature to the IoT device. From there, the device finalizes the signature generation by using its private key and choosing the message. It forwards the final signature to the IoT platform.

Outsourced and final signatures are constructed as in the threshold case, based on polynomial fractions defined over the attributes of the device and of the access policy. By doing so, if the device holds less attributes than a certain threshold, the signature verification will fail and the device will not have

Fig. 1. CHARIOT protocol overview



access to the platform. Contrary to [4], the signature generation does not require the use of dummy attributes into polynomials to reach correctness of the verification process. Moreover, signatures embed Groth-Sahai proof systems allowing the device to implicitly prove that it can decrypt a ciphertext corresponding to the ABE scheme [7].

Once receiving an access request from the IoT device, the platform executes Verify to check the validity of the device's signature and thus its right for access, using the public parameters, the current policy and the chosen message. If the result is positive, meaning that the device has the required attributes satisfying the policy, then it is authorized for access. The verification phase works thanks to the correctness of the ABE scheme [7] and to the perfect completeness, soundness and composable zero-knowledge of the Groth-Sahai proofs.

Figure 1 illustrates the CHARIOT protocol where a trusted attribute authority, an IoT platform, a cloud server and an IoT device participate.

D. Construction

The CHARIOT construction is made of six algorithms:

$\mathsf{Setup}(\lambda, \mathcal{P}, n) \rightarrow (params, msk)$**.** On inputs the security parameter λ, an attribute universe $\mathcal{P}$ and an integer n that is an upper bound on the size of threshold policies, the algorithm outputs the public parameters $params$ (which contain $(\lambda, \mathcal{P}, n)$) and the master secret key msk (for the trusted attribute authority) as follows:

The algorithm first chooses two cyclic groups $\mathbb{G}, \mathbb{G}_T$ of prime order $p > 2^\lambda$ with an efficiently computable bilinear map $e : \mathbb{G} \times \mathbb{G} \rightarrow \mathbb{G}_T$. Let g, h be two generators of $\mathbb{G}$ and $H : \{0,1\}^* \rightarrow \{0,1\}^k$ be a collision-resistant hash function for some $k \in \mathbb{N}$. Let τ be a HMAC that, given a key K, sends an attribute $at \in \mathcal{P}$ onto an element $\tau(K, at) \in \mathbb{Z}_p^*$ such that all output values are different. The algorithm then picks $\alpha, \beta, \gamma \in_R \mathbb{Z}_p^*$ and computes $u = g^\beta$, $v_i = g^{\frac{\alpha}{\gamma^i}}$ and $h_i = h^{\alpha\gamma^i}$ for $i \in [0, n]$. It generates the Groth-Sahai CRS by first choosing two generators g_1, g_2 of $\mathbb{G}$. Then, it defines the vectors $\vec{g}_1 = (g_1, 1, g)^\top$ and $\vec{g}_2 = (1, g_2, g)^\top$. For $i \in [0, k]$, it picks $\xi_{i,1}, \xi_{i,2} \in_R \mathbb{Z}_p$ and defines the vector $\vec{g}_{3,i} = (\vec{g}_1)^{\xi_{i,1}} \cdot (\vec{g}_2)^{\xi_{i,2}} = (g_1^{\xi_{i,1}}, g_2^{\xi_{i,2}}, g^{\xi_{i,1}+\xi_{i,2}})^\top$. Exponents $\{\xi_{i,1}, \xi_{i,2}\}_{i\in[0,k]}$ can then be discarded since they are no longer needed.

Finally, the algorithm sets the public parameters $params = (\lambda, \mathcal{P}, n, p, \mathbb{G}, \mathbb{G}_T, e, g, h, u, \{v_i\}_{i\in[0,n]}\{h_i\}_{i\in[0,n]}, \vec{g}_1, \vec{g}_2, \{\vec{g}_{3,i}\}_{i\in[0,k]}, H, \tau)$ and the master secret key $msk = (\alpha, \beta, \gamma)$.

$\mathsf{KeyGen}(params, msk, \Omega) \rightarrow (osk_\Omega, sk_\Omega, sk_{PT})$**.** On inputs the public parameters $params$, the master secret key msk, an attribute set $\Omega \subset \mathcal{P}$, the trusted attribute authority outputs the outsourcing key osk_Ω (for the cloud server), the private key sk_Ω (for the IoT device) and the secret key sk_{PT} (for the IoT platform) as follows:

Let K be a random key to be shared between the device and the platform, that will be used to hash the attributes. Let $\Omega \subset \mathcal{P}$ be an attribute set. The authority picks $\beta_1 \in_R \mathbb{Z}_p^*$ and sets $\beta_2 = \beta + \beta_1$. It then chooses $r \in_R \mathbb{Z}_p^*$ and computes $g^{\frac{r}{\gamma+\tau(K,at)}}$ for $at \in \Omega$, $h^{r\gamma^i}$ for $i \in [1, n-1]$, $h^{(r-\beta_2)\gamma^n}$, g^{β_1} and $h^{\beta_1\gamma^n}$.

The authority sets the outsourcing key $osk_\Omega = (\{g^{\frac{r}{\gamma+\tau(K,at)}}, \tau(K, at)\}_{at\in\Omega}, \{h^{r\gamma^i}\}_{i\in[1,n-1]}, h^{(r-\beta_2)\gamma^n}, g^{\beta_1})$ for the cloud server, the private key $sk_\Omega = (h^{\beta_1\gamma^n}, K)$ for the IoT device and the secret key $sk_{PT} = K$ for the IoT platform.

$\mathsf{Request}(\Gamma, sk_\Omega) \rightarrow \tilde{\Gamma}$**.** On inputs a threshold signing policy $\Gamma = (t, S)$ where the set $S \subset \mathcal{P}$ has $|S| = s \leq n$ attributes and $1 \leq t \leq s$, and the private key sk_Ω, the IoT device hashes each attribute $at \in S$ with τ resulting into $\tau(K, at)$, and creates the HMAC-hashed set $\tilde{S}$ containing the values $\tau(K, at)$ for all $at \in S$. It sets the HMAC-hashed threshold signing policy $\tilde{\Gamma} = (t, \tilde{S})$ and forwards it to the cloud server.

$\mathsf{Sign}_{out}(params, osk_\Omega, \tilde{\Gamma}) \rightarrow \sigma'$**.** On inputs the public parameters $params$, the outsourcing key osk_Ω and an HMAC-hashed threshold signing policy $\tilde{\Gamma} = (t, \tilde{S})$ where $\tilde{S}$ is the HMAC-hashed set of $S \subset \mathcal{P}$ and $1 \leq t \leq s \leq n$, the cloud server outputs an outsourced signature σ' as follows:

The cloud server returns 1 if $|\Omega \cap S| < t$; otherwise, it finds a subset $\Omega_S \subset \Omega \cap S$ such that $|\Omega_S| = t$. The cloud server works on the HMAC-hashed sets to verify the number of attributes contained in the intersection, since it should not get any information about the attributes in Ω and S except that there are at least t matching attributes.

For all $at \in \Omega_S$, the cloud server then runs the algorithm $\mathsf{Aggregate}(\{g^{\frac{r}{\gamma+\tau(K,at)}}, \tau(K, at)\}_{at\in\Omega_S}) = g^{\frac{r}{\prod_{at\in\Omega_S}(\gamma+\tau(K,at))}} = g^{\frac{r}{F_{\Omega_S}(\gamma)}} = T_1$. Let the polynomial $F_{S\setminus\Omega_S}(\gamma)$ be as $F_{S\setminus\Omega_S}(\gamma) = \prod_{at\in S\setminus\Omega_S}(\gamma + \tau(K, at)) = \sum_{i=0}^{s-t}\gamma^i b_i$ such that $b_{s-t} = 1$. Then, it computes $T_2' = h^{(r-\beta_2)\gamma^n} \cdot \prod_{i=0}^{s-t-1}(h^{r\gamma^{i+n-s+t}})^{b_i}$ that is possible given the public parameters $params$ and the outsourcing key osk_Ω. The obtained values T_1 and T_2' should satisfy the equality:

$$e(T_2', v_{n-s+t}^{-1}) \cdot e(T_1, h^{\alpha F_S(\gamma)}) = e(u \cdot g^{\beta_1}, h_{s-t})$$

Then, it picks $r_1, s_1, r_2, s_2 \in_R \mathbb{Z}_p$ and computes $\vec{C}'_{T_1} = (1, 1, T_1)^\top \cdot (\vec{g}_1)^{r_1} \cdot (\vec{g}_2)^{s_1}$ and $\vec{C}'_{T_2} = (1, 1, T_2')^\top \cdot (\vec{g}_1)^{r_2} \cdot (\vec{g}_2)^{s_2}$.

Let $\theta \in \mathbb{G}$ with commitment $\vec{C}'_\theta = (1,1,\theta)^\top \cdot (\vec{g}_1)^{r_\theta} \cdot (\vec{g}_2)^{s_\theta}$ for $r_\theta, s_\theta \in_R \mathbb{Z}_p$, which takes $\theta = h_{s-t}$ and proves the following:

$$e(T_1, H_S) = e(u \cdot g^{\beta_1}, \theta) \cdot e(T'_2, v_{n-s+t}) \quad (1)$$

$$e(g, \theta) = e(g, h_{s-t}) \quad (2)$$

where $H_S = h^{\alpha F_S(\gamma)} = h^{\alpha \prod_{at \in S}(\gamma + \tau(K,at))}$. Eqs. 1 and 2 are called proofs $\vec{\pi}'_1$ and $\vec{\pi}'_2$ respectively, and are given by $\vec{\pi}'_1 = (H_S^{r_1} \cdot (u g^{\beta_1})^{-r_\theta} \cdot v_{n-s+t}^{-r_2}, H_S^{s_1} \cdot (u g^{\beta_1})^{-s_\theta} \cdot v_{n-s+t}^{-s_2}, 1)^\top$ and $\vec{\pi}'_2 = (g^{r_\theta}, g^{s_\theta}, 1)^\top$. It also computes $g^{\beta_1 r_\theta}$ and $g^{\beta_1 s_\theta}$.

Finally, the cloud server sets the outsourced signature $\sigma' = (\vec{C}'_{T_1}, \vec{C}'_{T_2}, \vec{C}'_\theta, \vec{\pi}'_1, \vec{\pi}'_2, T'_2, H_S, g^{\beta_1 r_\theta}, g^{\beta_1 s_\theta})$ and forwards it to the IoT device.

$\mathsf{Sign}(params, sk_\Omega, \mathcal{M}, \sigma') \rightarrow \sigma$**.** On inputs the public parameters $params$, the private key sk_Ω, a message $\mathcal{M}$ and an outsourced signature σ', the IoT device outputs a signature σ as follows:

Given T'_2 from the outsourced signature σ' and $h^{\beta_1 \gamma^n}$ from the private key sk_Ω, the IoT device computes $T_2 = T'_2 \cdot h^{\beta_1 \gamma^n} = h^{(r-\beta_2)\gamma^n} \cdot \prod_{i=0}^{s-t-1} (h^{r\gamma^{i+n-s+t}})^{b_i} \cdot h^{\beta_1 \gamma^n} = h^{(r-\beta)\gamma^n} \cdot \prod_{i=0}^{s-t-1} (h^{r\gamma^{i+n-s+t}})^{b_i}$. The obtained values T_1 and T_2 should satisfy the equality:

$$e(T_2, v_{n-s+t}^{-1}) \cdot e(T_1, h^{\alpha F_S(\gamma)}) = e(u, h_{s-t}) \quad (3)$$

Thereafter, it computes $M = m_1 \cdots m_k = H(\mathcal{M}) \in \{0,1\}^k$. It uses M to form a message-specific Groth-Sahai CRS $\mathbf{g_M} = (\vec{g}_1, \vec{g}_2, \vec{g}_{3,M})$. More specifically, for all $i \in [0,k]$, $\vec{g}_{3,i}$ is parsed as $(g_{X,i}, g_{Y,i}, g_{Z,i})^\top$ and the device sets $\vec{g}_{3,M} = (g_{X,0} \cdot \prod_{i=1}^k g_{X,i}^{m_i}, g_{Y,0} \cdot \prod_{i=1}^k g_{Y,i}^{m_i}, g_{Z,0} \cdot \prod_{i=1}^k g_{Z,i}^{m_i})^\top$. The device then generates the Groth-Sahai commitments to the values T_1 and T_2 using $\mathbf{g_M}$. It picks $t_1, t_2 \in_R \mathbb{Z}_p$ and computes:

$$\vec{C}_{T_1} = \vec{C}'_{T_1} \cdot (\vec{g}_{3,M})^{t_1}$$

$$\vec{C}_{T_2} = \vec{C}'_{T_2} \cdot (1,1,h^{\beta_1 \gamma^n})^\top \cdot (\vec{g}_{3,M})^{t_2}$$

Then, it generates the NZIK proof that the pair of committed variables (T_1, T_2) satisfies the pairing-product in Eq. 3. To do so, let the commitment $\vec{C}_\theta = \vec{C}'_\theta \cdot (\vec{g}_{3,M})^{t_\theta} = (1,1,\theta)^\top \cdot (\vec{g}_1)^{r_\theta} \cdot (\vec{g}_2)^{s_\theta} \cdot (\vec{g}_{3,M})^{t_\theta}$ for $t_\theta \in_R \mathbb{Z}_p$, which takes $\theta = h_{s-t}$ and proves the following:

$$e(T_1, H_S) = e(u, \theta) \cdot e(T_2, v_{n-s+t}) \quad (4)$$

$$e(g, \theta) = e(g, h_{s-t}) \quad (5)$$

where $H_S = h^{\alpha F_S(\gamma)}$. Eqs. 4 and 5 are called proofs $\vec{\pi}_1$ and $\vec{\pi}_2$ respectively, and are given by:

$$\vec{\pi}_1 = \vec{\pi}'_1 \cdot (g^{-\beta_1 r_\theta}, g^{-\beta_1 s_\theta}, H_S^{t_1} \cdot u^{-t_\theta} \cdot v_{n-s+t}^{-t_2})^\top$$

$$\vec{\pi}_2 = \vec{\pi}'_2 \cdot (1,1,g^{t_\theta})^\top$$

Finally, the IoT device sets the signature $\sigma = (\vec{C}_{T_1}, \vec{C}_{T_2}, \vec{C}_\theta, \vec{\pi}_1, \vec{\pi}_2)$ and sends it to the IoT platform.

$\mathsf{Verify}(params, sk_{PT}, \mathcal{M}, \sigma, \Gamma) \rightarrow 0/1$**.** On inputs the public parameters $params$, the secret key sk_{PT}, a message $\mathcal{M}$, a signature σ and a threshold policy $\Gamma = (t, S)$, the IoT platform outputs 0 if the signature is valid and 1 otherwise.

The IoT platform computes $M = m_1 \cdots m_k = H(\mathcal{M})$, forms the vector $\vec{g}_{3,M} = (g_{X,0} \cdot \prod_{i=1}^k g_{X,i}^{m_i}, g_{Y,0} \cdot \prod_{i=1}^k g_{Y,i}^{m_i}, g_{Z,0} \cdot \prod_{i=1}^k g_{Z,i}^{m_i})^\top$, and sets $H_S = h^{\alpha F_S(\gamma)} = h^{\alpha \prod_{at \in S}(\gamma + \tau(K,at))}$ where $K = sk_{PT}$. Let $\vec{\pi}_j = (\pi_{j,1}, \pi_{j,2}, \pi_{j,3})^\top$ for $j \in \{1,2\}$. It returns 0 if and only if:

$$E(H_S, \vec{C}_{T_1}) = E(u, \vec{C}_\theta) \cdot E(v_{n-s+t}, \vec{C}_{T_2}) \cdot E(\pi_{1,1}, \vec{g}_1) \cdot E(\pi_{1,2}, \vec{g}_2) \cdot E(\pi_{1,3}, \vec{g}_{3,M})$$

$$E(g, \vec{C}_\theta) = E(g, (1,1,h_{s-t})) \cdot E(\pi_{2,1}, \vec{g}_1) \cdot E(\pi_{2,2}, \vec{g}_2) \cdot E(\pi_{2,3}, \vec{g}_{3,M})$$

Correctness. For any $\lambda, n \in \mathbb{N}$, any universe $\mathcal{P}$, any public parameters and master secret key $(params, msk) \leftarrow \mathsf{Setup}(\lambda, \mathcal{P}, n)$, any set $\Omega \subset \mathcal{P}$, any threshold policy $\Gamma = (t, S)$ where $1 \leq t \leq |S| \leq n$, and any message $\mathcal{M}$, it is required that $\mathsf{Verify}(params, sk_{PT}, \mathcal{M}, \mathsf{Sign}(params, sk_\Omega, \mathcal{M}, \mathsf{Sign}_{out}(params, osk_\Omega, \mathsf{Request}(\Gamma, sk_\Omega))), \Gamma) = 0$ whenever $(osk_\Omega, sk_\Omega, sk_{PT}) \leftarrow \mathsf{KeyGen}(params, msk, \Omega)$ and $|\Omega \cap S| \geq t$. The correctness is demonstrated based on the one for Groth-Sahai proofs, including the correctness of Eq. 3 that is given in the full version of the paper [11].

E. Security Analysis

The unforgeability proof relies on the Decisional LINear (DLIN) problem [4] related to the group $\mathbb{G}$ and on the Augmented Multi-Sequence of Exponents Computational Diffie-Hellman (aMSE-CDH) problem [4], [6] related to the group pair $(\mathbb{G}, \mathbb{G}_T)$. The privacy proof relies on the DLIN problem related to the group $\mathbb{G}$ [4]. We let the reader refer to the full version of the paper [11] for the proof sketches of CHARIOT.

IV. Performance Analysis

A. Performance Comparison

We compare the computational and storage costs between our CHARIOT scheme and the original Threshold ABS scheme [4]. In the following tables, "$\mathrm{Exp}_\mathbb{G}$", "$\mathrm{Mult}_\mathbb{G}$" and "$\mathrm{Pair}_{\mathbb{G}_T}$" denote exponentiation and multiplication in $\mathbb{G}$ and pairing operation in $\mathbb{G}_T$ respectively. "n.a." means "not available". Let n be the upper bound on the size of the policies and k be the parameter used for the Groth-Sahai proofs. Let Ω be the device's attribute set, S be the signing attribute set such that $s = |S|$, t be the threshold value and $\Omega_S = \Omega \cap S$ such that $|\Omega_S| = t$.

In Table I, let d be the number of dummy attributes such that $d = n+t-1-s \leq n-1$ as in [4]. $''-''$ means no computation of the given operation. Using dummy attributes in [4] increases the number of multiplications and exponentiations in $\mathbb{G}$ during the signature generation. By removing dummy attributes, up to $n-1$ modular multiplications and exponentiations are cut for the CHARIOT signing phase. Then, in CHARIOT, the computations done by a device are limited to the Groth-Sahai proofs' ones; all other calculations are delegated to the cloud server. Therefore, with the assistance of a cloud server, the device is relieved from most of the signing computations.

TABLE I
COMPUTATIONAL COST

	Threshold ABS [4]		CHARIOT	
	$\text{Mult}_{\mathbb{G}}$	$\text{Exp}_{\mathbb{G}}$	$\text{Mult}_{\mathbb{G}}$	$\text{Exp}_{\mathbb{G}}$
Setup	-	$2n+3k+5$	-	$2n+3k+6$
KeyGen	-	$\lvert\Omega\rvert+n$	-	$n+\lvert\Omega\rvert+2$
Sign_{out}	n.a.	n.a.	$s-t+17$	$2s+t^2-t+32$
Sign	$d+s$ $-t+17$ $+3k+5$	$d+2s$ $+t^2-t+32$ $+3k+4$	$3k+5$	$3k+4$
Verify	$3k$	$s+3k+1$	$3k$	$s+3k+1$

TABLE II
CHARIOT STORAGE COST COMPARED TO [4]

$params$	msk	osk_Ω	sk_Ω	σ
$-d-1$	$+1$	$+\lvert\Omega\rvert+n+3$	$-\lvert\Omega\rvert-n+2$	$=$

In Table II, only the storage difference in CHARIOT compared to the ABS scheme in [4] is taken into account: an element $+1$ (resp. an element -1) in one cell means that there is one extra element (resp. there is one less element) in the CHARIOT protocol compared to Herranz et al.'s protocol. $''=''$ in a given column denotes that storage is identical in the two protocols regarding the component linked to this column. The number d of dummy attributes is equal to $n-1$ as in [4]. Hence, in [4], a set of $n-1$ dummy attributes should be stored in $params$ in order to let the device generate its signature, while in CHARIOT, such storage cost is saved. An extra secret element is required in msk in CHARIOT to enable the outsourcing of signing computations. The outsourcing key osk_Ω in CHARIOT is mainly the private key sk_Ω in [4]. The device in CHARIOT simply receives an extra element from the shared secret and the key for the HMAC τ as its private key sk_Ω. Signatures are of equal size in Herranz et al.'s ABS [4] and CHARIOT schemes, and independent of the number of attributes.

B. Experimental Study

We implement CHARIOT in Charm [12] based on Python language. We set two different configurations[2], one for the trusted attribute authority, the cloud server and the IoT platform, and another one for the IoT device. We assume that the attribute authority, the cloud server and the platform have similar resources, while the device has much less resources than these three parties.

Policies are supposed to contain up to 30 attributes regarding real scenarios [13], [14]. Hence, we choose an upper bound n equal to 30, a policy set S with $s=15$ attributes, a threshold t equal to 13 and a device's attribute set Ω with 20 attributes. The parameter k is chosen regarding the bit-size of the hashed message. Since the message to be signed is public, the hash function does not require to be cryptographic but rather collision-resistant. We also suppose that the dictionary of messages picked by IoT devices is of finite and moderated size. Therefore, we evaluate k with the values 10, 20 and 40 in order to avoid collisions with high probability. The given timings are an average from 10 rounds of the CHARIOT protocol.

[2]The experiments are tested on a processor Intel Core i5-2500 CPU @3.30GHz $\times 4$ with RAM 16GiB and OS Linux Ubuntu 14.04 LTS (Configuration 1 for the attribute authority, cloud server and platform) and on a processor Genuine Intel(R) U2300 CPU @1.20GHz with RAM 2GiB and OS Linux Ubuntu 17.10 (Configuration 2 for the device).

TABLE III
TIMINGS IN MILLISECONDS

		$k=10$	$k=20$	$k=40$
Configuration 1 (attribute authority, cloud server and platform)	Setup	143.77	190.67	284.54
	KeyGen	76.19	75.47	75.17
	Sign_{out}	265.07	272.69	271.55
	Verify	65.66	108.93	194.12
Configuration 2 (device)	Request	0.16	0.16	0.17
	Sign	182.55	322.01	601.6

In Table III, the setup phase is costly but should be executed only once. It largely depends on the upper bound n on the size of threshold policies and on the parameter k used for Groth-Sahai proofs. The key generation phase is also performed only once by the trusted attribute authority, and is relatively fast since it only relies on n that should not exceed 30 in an IoT environment.

By outsourcing the signature generation to a cloud server, we speed up by three times the computational timing. Indeed, the cloud server approximately needs 270 milliseconds to run the algorithm Sign_{out} while the device would require 870 milliseconds to do the same (when setting Configuration 2 as for the device). We recall that the algorithm Sign_{out} does not depend on the parameter k, and thus does not variate with it. Most of the signature generation is thus done by the cloud server running the algorithm Sign_{out}. The finalization of the signature generation is accomplished by the device by running the algorithm Sign and remains less than 870 milliseconds (resulting from running the algorithm Sign_{out} with Configuration 2). However, timings for signature finalization and verification increase with the parameter k as these two phases largely rely on such parameter. Hence, k should not exceed 40 in order to keep the advantage of outsourcing the signature generation to a cloud server.

In the CHARIOT protocol, the parameter k refers to the bit-size of the hashed message. This parameter k should be selected such that collisions are highly avoided. If the dictionary of messages to be signed is really small, then $k=10$ is optimal. Otherwise, if the dictionary is slightly bigger, then $k=20$ is still a reasonable choice. In addition, signature finalization and verification take similar timings to be performed since they are composed of the same kinds of computations. The difference comes from the constant number (equal to 27) of pairing operations carried out when checking the validity of the signature. Because of the number of pairing computations required for verification, the Groth-Sahai proof systems appear to be inefficient. Blazy et al. [15] propose a significant reduction of the cost of Groth-Sahai proof systems by using batch verification techniques. The authors improve

the verification cost up to four times the number of pairings per proof verification. We can integrate their batch verification techniques into the CHARIOT protocol to improve the timings of the verification phase.

C. Related Work

Maji et al. [16] introduce the notion of ABS. Subsequent works on ABS are given in [3], [17], [18] where trade-off between efficiency and security is not optimal. More recently, Herranz et al. [4] suggest an ABS construction with threshold policies and constant-size signatures, which requires the presence of dummy attributes following the technique from Dynamic Threshold Public-Key Encryption [19].

The problem of securely outsourcing expensive computations is studied in [20]–[22]. However, alleviating computational burdens induced by signature generation is not possible. Server-aided signature schemes [23], [24] aim to decrease the computational cost due to exponentiation calculations by outsourcing the latter to a server, but access control management based on credentials is not enabled. Mediated cryptographic protocols [25], [26] require a partially trusted on-line server but cannot be used to extend ABS into an outsourced ABS. More recently, Chen et al. [5] present two outsourced ABS schemes. Yet, the cloud server must know the attributes of the signers and of the signing policies in plain in order to proceed, compromising privacy.

V. Conclusion

In this paper, we propose CHARIOT, a new server-aided access control protocol in IoT with cloud server assistance, constant-size signature and no dummy attributes. Outsourcing most of the calculations to a cloud server for the signature generation relieves the workload at the device's side. Removing the presence of dummy attributes implies a more efficient and practical ABS scheme compared to existing ones. Moreover, privacy and secure identity management of the involved parties are guaranteed. These contributions make our scheme suitable for securely authorizing devices with constrained resources to access an IoT platform.

References

[1] J. Gubbi, R. Buyya, S. Marusic, and M. Palaniswami, "Internet of things (iot): A vision, architectural elements, and future directions," *Future Generation Computer Systems*, vol. 29, no. 7, pp. 1645–1660, 2013.

[2] J. Su, D. Cao, B. Zhao, X. Wang, and I. You, "ePASS: An expressive attribute-based signature scheme with privacy and an unforgeability guarantee for the internet of things," *Future Generation Computer Systems*, vol. 33, no. Supplement C, pp. 11 – 18, 2014.

[3] J. Li, M. H. Au, W. Susilo, D. Xie, and K. Ren, "Attribute-based signature and its applications," in *Proceedings of ASIACCS'10*, 2010, pp. 60–69.

[4] J. Herranz, F. Laguillaumie, B. Libert, and C. Ràfols, "Short attribute-based signatures for threshold predicates," in *Proceedings of CT-RSA'12*, 2012, pp. 51–67.

[5] X. Chen, J. Li, X. Huang, J. Li, Y. Xiang, and D. S. Wong, "Secure outsourced attribute-based signatures," *IEEE Transactions on Parallel and Distributed Systems*, vol. 25, no. 12, pp. 3285–3294, 2014.

[6] W. Susilo, G. Yang, F. Guo, and Q. Huang, "Constant-size ciphertexts in threshold attribute-based encryption without dummy attributes," *Information Sciences*, vol. 429, no. C, pp. 349–360, Mar. 2018.

[7] J. Herranz, F. Laguillaumie, and C. Ráfols, "Constant size ciphertexts in threshold attribute-based encryption," in *Proceedings of PKC'10*, 2010, pp. 19–34.

[8] J. Groth and A. Sahai, "Efficient non-interactive proof systems for bilinear groups," in *Proceedings of EUROCRYPT'08*, 2008, pp. 415–432.

[9] T. Malkin, I. Teranishi, Y. Vahlis, and M. Yung, "Signatures resilient to continual leakage on memory and computation," in *Proceedings of TCC'11*, 2011, pp. 89–106.

[10] B. Waters, "Efficient identity-based encryption without random oracles," in *Proceedings of EUROCRYPT'05*, 2005, pp. 114–127.

[11] C. Gritti, M. Önen, and R. Molva, "Chariot: Cloud-assisted access control for the internet of things," Cryptology ePrint Archive, Report 2018/632, 2018, https://eprint.iacr.org/2018/632.

[12] J. A. Akinyele, M. D. Green, and A. D. Rubin, "Charm: A framework for rapidly prototyping cryptosystems," Cryptology ePrint Archive, Report 2011/617, 2011, https://eprint.iacr.org/2011/617.

[13] X. Yao, Z. Chen, and Y. Tian, "A lightweight attribute-based encryption scheme for the internet of things," *Future Generation Computer Systems*, vol. 49, no. Supplement C, pp. 104 – 112, 2015.

[14] M. Ambrosin, A. Anzanpour, M. Conti, T. Dargahi, S. R. Moosavi, A. M. Rahmani, and P. Liljeberg, "On the feasibility of attribute-based encryption on internet of things devices," *IEEE Micro*, vol. 36, no. 6, pp. 25–35, Nov 2016.

[15] O. Blazy, G. Fuchsbauer, M. Izabachène, A. Jambert, H. Sibert, and D. Vergnaud, "Batch groth-sahai," in *Proceedings of ACNS'10*, 2010, pp. 218–235.

[16] H. K. Maji, M. Prabhakaran, and M. Rosulek, "Attribute-based signatures," in *Proceedings of CT-RSA'11*, 2011, pp. 376–392.

[17] J. Li and K. Kim, "Hidden attribute-based signatures without anonymity revocation," *Information Science*, vol. 180, no. 9, pp. 1681–1689, 2010.

[18] T. Okamoto and K. Takashima, "Efficient attribute-based signatures for non-monotone predicates in the standard model," in *Proceedings of PKC'11*, 2011, pp. 35–52.

[19] C. Delerablée and D. Pointcheval, "Dynamic threshold public-key encryption," in *Proceedings of CRYPTO'08*, 2008, pp. 317–334.

[20] M. J. Atallah and J. Li, "Secure outsourcing of sequence comparisons," *Information Security*, vol. 4, no. 4, pp. 277–287, 2005.

[21] M. J. Atallah and K. B. Frikken, "Securely outsourcing linear algebra computations," in *Proceedings of ASIACCS'10*, 2010, pp. 48–59.

[22] C. Wang, K. Ren, and J. Wang, "Secure and practical outsourcing of linear programming in cloud computing," in *Proceedings of INFOCOM'11*, 2011, pp. 820–828.

[23] M. Jakobsson and S. Wetzel, "Secure server-aided signature generation," in *Proccedings of PKC'01*, 2001, pp. 383–401.

[24] K. Bicakci and N. Baykal, "Server assisted signatures revisited," in *Proceedings of CT-RSA'04*, 2004, pp. 143–156.

[25] D. Boneh, X. Ding, G. Tsudik, and C. M. Wong, "A method for fast revocation of public key certificates and security capabilities," in *Proceedings of SSYM'01*, 2001.

[26] D. Boneh, X. Ding, and G. Tsudik, "Fine-grained control of security capabilities," *ACM Transactions on Internet Technology*, vol. 4, no. 1, pp. 60–82, 2004.