Using Attribute-Based Encryption on IoT Devices with instant Key Revocation

Marten Fischer, Alfred Scheerhorn, Ralf Tönjes
University of Applied Sciences Osnabrück
Germany
E-mail: {m.fischer,a.scheerhorn,r.toenjes}@hs-osnabrueck.de

***Abstract*—The Internet of Things (IoT) relies on sensor devices to measure real-world phenomena in order to provide IoT services. The sensor readings are shared with multiple entities, such as IoT services, other IoT devices or other third parties. The collected data may be sensitive and include personal information. To protect the privacy of the users, the data needs to be protected through an encryption algorithm. For sharing cryptographic cipher-texts with a group of users Attribute-Based Encryption (ABE) is well suited, as it does not require to create group keys. However, the creation of ABE cipher-texts is slow when executed on resource constraint devices, such as IoT sensors. In this paper, we present a modification of an ABE scheme, which not only allows to encrypt data efficiently using ABE but also reduces the size of the cipher-text, that must be transmitted by the sensor. We also show how our modification can be used to realise an instantaneous key revocation mechanism.**

I. Introduction

The Internet of Things (IoT) has great potential to bridge the gap between the virtual world and the real world. In IoT, sensors are deployed in the real-world to measure real-world physical phenomena. This information can then be processed by services or applications to help users to make better decisions or to influence the real world directly via actuators. In general, IoT sensors are small, resource constraint devices, with low memory and low computational power. Furthermore, it is most likely, that many of these IoT sensors are battery powered, meaning long-running operations should be avoided to extend the life-span or to reduce maintenance intervals respectively. The provider of an IoT sensor network might want to protect the data gathered by the sensors due to privacy reasons or as a part of his business model. For this, he/she will utilise cryptographic methods. In order to avoid the need for re-encrypting the data for each data user, a public key cryptographic scheme, such as RSA [9], combined with a group key can be used. Recently, a cryptographic scheme more suited to share secret cipher-texts with a group of users has emerged, namely Attribute-Based Encryption (ABE). In ABE a public key is a set of attributes describing the intended receiver. A user possessing a (sub-)set of these attributes is able to decrypt the cipher-texts. This means, an ABE cipher-text is not encrypted for a specific user or a user's private key, but to publicly available attributes. To ensure a user possesses a certain attribute a new entity called Attribute Authority (AA) is introduced. Data users authenticate at the AA and request a private key corresponding to an attribute-set, with which decryption is possible. However, the creation of an ABE cipher-text is a complex task, not well suited for resource constraint IoT devices. The complexity is directly proportional to the number of attributes used during encryption. In this paper, we present a modification to an existing ABE scheme, which reduces the computational effort on the IoT device significantly. At the same time, the size of the cipher-text is reduced as well, requiring less communication efforts to transfer encrypted sensor readings from the sensor to a network/cloud storage. Both measures enable the usage of ABE group encryption on IoT sensors.

A major problem in ABE is the key revocation. Unlike in other public key cryptographic schemes, attributes may be shared among multiple users, meaning the revocation of one attribute may effect multiple users. Since an encrypter uses only publicly available attributes for encryption, he/she cannot be aware of any revocations that may have occurred. We show that our modification can be used to employ an efficient revocation mechanism. The mechanism can be configured by implementing simple rules into the encryption process.

The remainder of this paper is structured as follows: Section II presents related work for key revocation techniques in ABE and ABE based solutions for the IoT. Section III summarises the workflow and algorithms of an ABE scheme (CP-ABE), which is the basis for our modification. Our modification to the CP-ABE scheme is presented in IV. Section V compares the performance of the original ABE scheme and our modification. Section VI describes how our modification can be used to realise a key revocation mechanism. Finally, section VII concludes the paper.

II. Related Work

A. Key Revocation in ABE

Key revocation is a challenging task in ABE (and Identity-Based Encryption for that matter). Because in ABE cipher-texts are created for a group of users, revocation attempts might affect multiple users.

[10] proposed to attach an expiration date to each attribute to realise a key revocation mechanism. An attribute *student* would become for example *student2018*. A student with such attribute must update his private key periodically. The method has the disadvantage, that the revocation would take possibly a long time. Even extending it to *studentOct2018*, the revocation time can be up to 1 month long. Bethencourt et al. [3] showed

978-1-5386-9151-9/19/$31.00 ©2019 IEEE

how to construct an access tree to realise numerical comparison. They use the numerical comparison also to realise a sort of key revocation functionality by assigning an expiration value to each attribute.

A more instantaneous approach proposed by [7] utilises proxy-re-encryption. Proxy-re-encryption is a technique, that allows modifying a cipher-text, without learning something about the corresponding plain-text. Here, a central storage system (e.g. cloud storage) provides a directory service with all cipher-texts. Upon a revocation event, this central system generates new private key components for all non-revoked users and distributes them. Afterwards all cipher-texts are re-encrypted using the proxy-re-encryption technique. In this approach, the revocation time is drastically reduced. However, a lot of computational effort is required to re-encrypt all cipher-texts, especially when the system is running for a long time.

The authors of [13] also use proxy-re-encryption to realise a revocation mechanism. Here, a synthetical attribute called *delegation attribute* is used. As before, a central storage system provides a directory service with all cipher-texts. When a user requests a cipher-text, the central system embeds the delegation attribute into the cipher-text in such a way, that only non-revoked users can decrypt the cipher-text. A cipher-text is returned in both cases, either a valid one or a cipher-text, which cannot be decrypted by any user in the system. It remains unclear why the authors decided to design the system this way, instead of simply replying with an error message. In this scheme, a data user must authenticate at two entities, the AA to get the private user key and at the central storage system, to get the cipher-text. In turn, a user database needs to be synchronized between both entities.

None of the schemes addresses the problem of computational costs for resource constraint devices.

B. ABE for Resource Constraint Devices

The algorithms involved in ABE are not well suited to be performed on resource constraint devices. Ambrosin et al. did a performance analysis using a Samsung Galaxy smart-phone with 1.2 GHz CPU and 1GB RAM [2]. Their results showed, that to encrypt a cipher-text with 30 attributes requires up to 30 seconds, depending on the security level. Considering IoT devices, the performance figures of a smart-phone seem high and much more time would be required to encrypt a message.

In [6], Green et al. try to reduce the required performance in order to decrypt an ABE cipher-text. Their goal was to enable smart-phones to access and decrypt ABE cipher-texts efficiently and in a timely manner. Their approach was to use a near-edge proxy, which does the ”heavy” work involved in the decryption process. The proxy performs a transformation from hard to process ABE cipher-text into an easier cipher-text. The smart-phone performs the final decryption to recover the plain-text. Of course, the proxy is at no time able to access the plain-text.

The authors of [8] tried to speed up the encryption process by pre-computing and storing tuples. These tuples are later used to define a number of polynomials with some specific properties. The obvious drawback in this approach is the additional memory requirement. The number of tuples created during the pre-computation process is directly proportional to the number of attributes in a defined access policy. Modifications on the access policy require to re-run the pre-computation process. Another drawback is the size of the cipher-text, which increases with the number of attributes.

Touati et al. [12] propose to distribute calculations (mainly exponentiations) of the ABE encryption algorithm to neighbouring IoT nodes. However, their approach requires, that each IoT node shares pairwise keys with at least two unconstraint trusted nodes in the neighbourhood. Our approach presented in this paper follows the similar goal: distributing the exponentiation calculations to an unconstraint entity.

In contrast to the scheme by Touati, our approach requires only one unconstraint entity and no pairwise keys need to be exchanged beforehand. The size of the cipher-text transferred from the IoT device to the entity is reduced significantly. In addition, none of the approaches to enable ABE for IoT devices has dealt with the problem of key revocation. Our approach supports a configurable key revocation mechanism, which can deny access to an encrypted message instantaneously.

III. Attribute-Based Encryption

In 1984, Shamir introduced a novel type of cryptographic scheme that enabled users to communicate securely without the need of exchanging private or public keys and without requiring the services of a Public-Key-Infrastructure (PKI) [11]. In his approach, instead of generating a random pair of private and public keys, the user chooses the identification (e.g. email address) of his communication partner as the public key. A Private Key Generator (PKG), holding a secret master key, computes the corresponding private key once and issues it to the user. Anyone who wants to send a cipher-text to a user can use the recipient's identification as the public key for encryption. The receiver uses the private key generated by the PKG to decrypt the cipher-text. The concept is depicted in Figure 1. Because the scheme uses the user's identity as the public key, it was named Identity-Based Encryption (IBE). The first fully-functional IBE scheme was developed by Boneh and Franklin 17 years later [4].

IBE was extended to Fuzzy Identity-Based Encryption (Fuzzy-IBE) by Sahai and Waters [10]. They envisioned to use biometric identities in an IBE scheme, to encrypt data based on a user's biometric properties, e.g. an iris scan. Because biometric measurements are noisy, the scheme must provide some degree of error-tolerance, to allow fuzzy attribute-sets, hence the name Fuzzy Identity-Based Encryption.

A second application was named **Attribute-Based Encryption (ABE)**. Here, the encrypter chooses a set of attributes in order to encrypt a message. Any user that has as identity a required number of these attributes is able to decrypt the cipher-text.

Later, two important derivates were developed. Both use an access tree (as defined in P-1) to describe an access

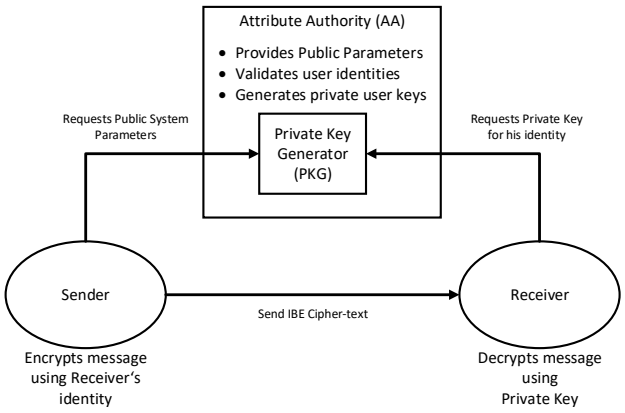

Fig. 1. Participants in an Identity-Based Encryption scheme

policy utilising attributes. The first derivate is called Key-Policy Attribute-Based Encryption (KP-ABE), where the access policy is associated with the private user key and the cipher-text is associated with an attribute-set [5]. In the second derivate, named Ciphertext-Policy Attribute-Based Encryption (CP-ABE), the situation is reversed. Here the private user key is associated with an attribute-set and the cipher-text is associated with the access policy. Our modification is based on the CP-ABE scheme. For this reason subsection III-B will describe CP-ABE in more detail. Before that, we state some preliminaries used in this and the following section.

A. Preliminaries

P-1 Access Tree: Let $\mathcal{T}$ be an access tree, where each leaf node represents an attribute and each non-leaf node represents a threshold gate. Let num_n be the number of child nodes for node n and k_n be the threshold value with $0 \leq k_n \leq num_n$. We define the function $I_{\mathcal{T}}(n)$ to return a unique identifier (an ordering of the children of node n) and $P_{\mathcal{T}}(n)$ to return the parent node of node n in $\mathcal{T}$. We denote a set of attributes ω that satisfies an access tree $\mathcal{T}$ with $\mathcal{T}(\omega) = 1$. The attribute of a leaf node n is denoted with a_n.

P-2 Bilinear Map: Let $\mathbb{G}_1$ and $\mathbb{G}_2$ be groups of prime order p and let g be a generator of $\mathbb{G}_1$. We say $\mathbb{G}_1$ has a bilinear map $e : \mathbb{G}_1 \times \mathbb{G}_1 \rightarrow \mathbb{G}_2$, if the following two conditions hold: (i) The mapping is bilinear, i.e. for all a, b we have $e(g^a, g^b) = e(g, g)^{\text{ab}}$ and (ii) $e(g, g) \neq 1$.

P-3 Random Oracle: A random oracle is a function, which maps each input to a uniformly chosen random output from its output domain. The output of a request has exactly one input, i.e. the function is injective.

P-4 The Lagrange Coefficient: $\triangle i, S(x)$ for $i \in \mathbb{Z}_p$ and a set S of elements in $\mathbb{Z}_p$ is defined as $\prod_{j \in S, i \neq j} \frac{x-j}{i-j}$.

B. Ciphertext-Policy Attribute-Based Encryption

The CP-ABE scheme was introduced by Bethencourt et al. in [3]. Here, the cipher-text embeds the access policy in form of an access tree $\mathcal{T}$ to describe which private keys can decrypt it. This means, that the data owner is capable to specify the access policies at the cipher-texts to state which data users will be able to decrypt them. Therefore the CP-ABE uses a more intuitive method compared to KP-ABE, as it is similar to a traditional access control model and allows a more versatile sharing of data among groups. The data user's private key is labelled with a set of descriptive attributes ω'. The CP-ABE scheme is composed of five algorithms: *Setup*, *KeyGen*, *Encrypt*, *Decrypt* and an optional *Delegate*, which are presented in the following. The scheme uses a bilinear map $\mathbb{G}_1$ as defined in P-2 and an access tree $\mathcal{T}$ as defined in P-1.

1) $(PK, MK) \leftarrow Setup()$: The *Setup* algorithm executed by the AA has no input and outputs the system's public key PK and the system's master key MK. The master key MK is kept secret by the AA and the public key is published to all users in the system.
2) $D \leftarrow KeyGen(MK, \omega')$: The *KeyGen* algorithm generates a private user key D. The algorithm takes as input the secret master key MK and the set of attributes ω', which describes the user. The private key components are randomised so that no two users can collude by combining their components.
3) $E \leftarrow Encrypt(PK, \mathcal{T}, M)$: The *Encrypt* algorithm encrypts a message $M \in \mathbb{G}_1$ using access policy $\mathcal{T}$. In addition, the algorithm takes as input the public key PK and returns cipher-text E.
4) $M \leftarrow Decrypt(D, E)$: The *Decrypt* algorithm uses the private key D and the cipher-text E to recover message M. This algorithm is only successful, if the attribute-set ω', used to create the private key D, satisfies the access policy $\mathcal{T}$ used in the *Encrypt* algorithm.
5) $\tilde{D} \leftarrow Delegate(D, \tilde{\omega})$: The *Delegate* algorithm allows a data user, that has already obtained a private key D for his attribute-set ω' from the AA, to create a new private key $\tilde{D}$, which is labelled with a subset of attributes $\tilde{\omega} \subseteq \omega'$. This algorithm is optional and not further considered in this paper.

IV. Ciphertext-Policy Attribute-Based Encryption for Resource Constraint IoT Devices

In this section, we show our modification to reduce the computational costs for the creation of CP-ABE cipher-texts on resource constraint IoT devices. In this modification, only a partial CP-ABE cipher-text is created on the resource constraint device itself (during the *Encrypt* algorithm) and completed later on an unconstraint entity called proxy in the *Convert* algorithm. In the following we describe the algorithms for our modification in detail:

1) $(PK, MK) \leftarrow Setup()$: The Setup algorithm chooses a bilinear group $\mathbb{G}_1$ of prime order p with generator g and three random exponents $\alpha, \beta, \gamma \in \mathbb{Z}_p$. The system's public parameters are $PK = (\mathbb{G}_1, g, h = g^\beta, b = g^\gamma, e(g, g)^\alpha)$ and the secret master key is $MK = (\alpha, \beta, \gamma)$.
2) $D \leftarrow KeyGen(MK, \omega')$: The KeyGen algorithm generates a private key for a user identified with the attribute-set ω'. For this, the algorithm generates the random

numbers r and $r_i \in \mathbb{Z}_p \forall i \in \omega'$. The private key D is generated as $D = (D' = g^{r/\beta}, D_i = \{g^r \cdot H(i)^{r_i}\}_{i \in \omega'}, D'_i = \{g^{r_i}\}_{i \in \omega'})$. ($H$ is a function modeled as random oracle as defined in P-3.) This algorithm is executed by the AA.

3) $E \leftarrow Encrypt(PK, \mathcal{T}, M)$: With this algorithm, the IoT device generates the cipher-text E by encrypting a message $M \in \mathbb{G}_1$ using the access tree $\mathcal{T}$. The algorithm chooses two random numbers $s, v \in \mathbb{Z}_p^*$ and creates the cipher-text as follows: $E = (\mathcal{T}, E' = Me(g,g)^{\alpha s}, Q = s/v, C = h^Q = h^{s/v}, V = b^v)$. E is then sent to the proxy.
4) $\hat{E} \leftarrow Convert(E)$: This algorithm is executed by the proxy and generates a polynomial q_n with degree d_n for each node $n \in \mathcal{T}$ with $\forall d_n : d_n = k_n - 1$, where k_n is the threshold value of node n. For the polynomial of the root node n_0, the algorithm sets $q_0(0)$ to $Q = s/v$ and d_0 other points at random to define it completely. For all other polynomials q_n the algorithm sets $q_n(0) = q_{P_{\mathcal{T}}(n)}(I_{\mathcal{T}}(n))$ and chooses d_n other points at random to define q_n completely. With these polynomials the cipher-text can be completed as $\hat{E} = (\mathcal{T}, E', C, V, \{E_n = g^{q_n(0)}\}_{n \in L}, E'_n = \{H(a_n)^{q_n(0)}\}_{n \in L})$.
5) $\hat{D} \leftarrow Grant(MK, V, D')$: This algorithm is performed by the AA in order to grant a data user access to a cipher-text. The algorithm takes as input the secret master key MK, V from cipher-text $\hat{E}$, and D' from the users private key. First it computes $\hat{V} = V^{1/\gamma} = g^{v\gamma/\gamma} = g^v$. Using $\hat{V}$ and the secret master key the algorithm can compute $\hat{D} = \hat{V}^{\alpha/\beta} \cdot D' = g^{v\alpha/\beta} g^{r/\beta} = g^{v\alpha/\beta + r/\beta}$. $\hat{D}$ is then returned back to the data user. In section VI we explain how to implement a key revocation mechanism into this algorithm.
6) $M \leftarrow Decrypt(\hat{E}, D, \hat{D})$: This algorithm decrypts the cipher-text $\hat{E}$, associated with an access structure $\mathcal{T}$, using the private key D to recover the message M. The algorithm uses the function $DecryptNode$ to process the access structure. The function takes as input the cipher-text E, the private key D and a node n. If n is a leaf node and i is an attribute contained in $\omega' \mid i = a_n$ the function works as follows:

$$\begin{aligned} DecryptNode(\hat{E}, D, n) &= \frac{e(D_i, E_n)}{e(D'_i, E'_n)} \\ &= \frac{e(g^r \cdot H(i)^{r_i}, h^{q_n(0)})}{e(g^{r_i}, H(i)^{q_n(0)})} \\ &= e(g,g)^{r q_n(0)} \end{aligned}$$

Otherwise, it returns NULL. If n is not a leaf node the algorithm proceeds as follows: For all nodes C that are children of n it calls the function $DecryptNode(\hat{E}, D, c \in C)$ and stores its values in V_c. If there is a subset $S \subseteq C$ with $|S| = k_n$ and $\forall c \in S : V_c \neq$ NULL, then it computes

$$\begin{aligned} V_n &= \prod_{c \in S} e(g,g)^{r \cdot q_{P_{\mathcal{T}}(c)}(I_{\mathcal{T}}(c))} \\ &= \prod_{c \in S} e(g,g)^{r \cdot q_c(0)} \\ &= e(g,g)^{r \cdot q_n(0)} \text{ (using the Lagrange coefficient, P-4)} \end{aligned}$$

and returns the result. If the algorithm calls the function recursively on the root node n_0, it gets $V_0 = e(g,g)^{r \cdot q_0(0)} = e(g,g)^{rs/v}$, since $q_0(0)$ is set to $Q = s/v$ in the *Encrypt* algorithm. Now the algorithm can recover the message

$$\begin{aligned} M &= \frac{E'}{(e(C, \hat{D})/V_0)} \\ &= \frac{Me(g,g)^{\alpha s}}{(\frac{e(h^Q, g^{(v\alpha/\beta + r/\beta)})}{e(g,g)^{rs/v}})} \\ &= \frac{Me(g,g)^{\alpha s}}{(\frac{e(g^{\beta s/v}, g^{(v\alpha/\beta + r/\beta)})}{e(g,g)^{rs/v}})} \\ &= \frac{Me(g,g)^{\alpha s}}{(\frac{e(g,g)^{\beta s/v \cdot (v\alpha/\beta + r/\beta)}}{e(g,g)^{rs/v}})} \\ &= \frac{Me(g,g)^{\alpha s}}{(\frac{e(g,g)^{\alpha s + rs/v}}{e(g,g)^{rs/v}})} \\ &= \frac{Me(g,g)^{\alpha s}}{e(g,g)^{\alpha s}} \\ &= M. \end{aligned}$$

The only algorithm executed by the IoT device is the *Encrypt* algorithm. The number of exponentiation is reduced to 3 instead of $2+2n$ (with n being the number of attributes used in the access policy) in the original CP-ABE. The size of the cipher-text is reduced by the same degree, 3 group elements instead of $2 + 2n$ in the original CP-ABE. The calculation of the cipher-text components E_n and E'_n is shifted to the proxy executing the *Convert* algorithm. The proxy needs to be a trusted entity using the provided access policy $\mathcal{T}$ when generating the cipher-text components E_n and E'_n. The proxy is never able to decrypt the cipher-text E' to recover message M. Each time a new value for v is chosen in the *Encrypt* algorithm, a new $\hat{D}$ is required, to decrypt a cipher-text. In order to get $\hat{D}$, the data user sends V and D' to the AA, which executes the *Grant* algorithm. Since the private key component $\hat{D}$ includes the random number r, which is individual for each user in the system, no two users can collude. That means a non-revoked user cannot share his $\hat{D}$ (the output of the *Grant* algorithm) with a revoked user. For scalability reasons, the *Grant* algorithm can be distributed over Sub-AAs by providing them with the secret master key, i.e. α/β and γ. Figure 2

summarises the message flow between the involved parties (without the *Setup* algorithm and without public or master keys).

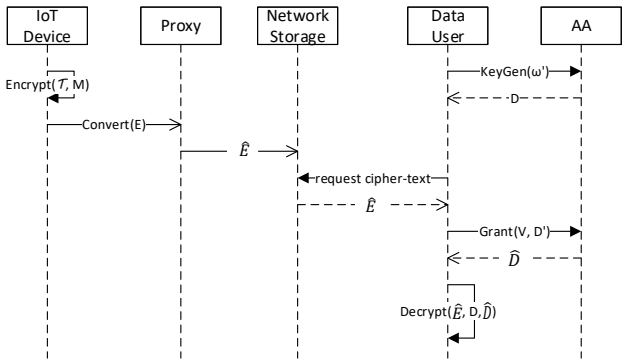

Fig. 2. Messageflow

Further improvement can be achieved, if the AA keeps a repository of the private key component D' for each user. In the subsequent *Grant* algorithm, a data user only needs to send V from cipher-text $\hat{E}$ to the AA. After authentication, the AA can lookup the D' required to compute $\hat{D}$. That way, an unnecessary transfer of D' (as part of the private key D) between data user and AA can be avoided.

V. Performance Evaluation

In this section, we compare the performance of the CP-ABE scheme, as introduced in [3] ("BSW07"), and the modification introduced in this paper ("IoT-ABE"). Both schemes were implemented in the programming language Python. The measurements of the CP-ABE scheme were done using the implementation provided in the Charm crypto framework [1]. The measurements focus on the algorithms required to create a CP-ABE cipher-text, namely the *Encrypt* (both schemes) and the *Convert* (this scheme) algorithms. The performance measurements were done on a Raspberry Pi 3 Model B v1.2, except for the *Convert* algorithm, which was executed on a laptop (2.6 GHz Intel Core i5). In this demonstration scenario, the Raspberry Pi represents the resource constraint IoT device and the laptop represents the proxy. The measurements were done in 10 runs with an increasing number of attributes in the access policy, ranging from 2 to 20. Each run was repeated 100 times. Figure 3 shows the mean execution time of all 100 repetitions.

The blue bars in figure 3 depict the execution time of the original CP-ABE *Encrypt* algorithm and show a linear growth with an increasing number of attributes, up to 1 second at 20 attributes. The green bars depict the execution time of the modified *Encrypt* algorithm executed on the Raspberry Pi. As expected, the execution time stays constant at about 0.022 seconds. The *Convert* algorithm is executed on the laptop and the execution time increases with the growing number of attributes as well. This can be seen at the orange bars in the figure. Since parts of the process to create an ABE cipher-text could be outsourced from the IoT device to a more powerful

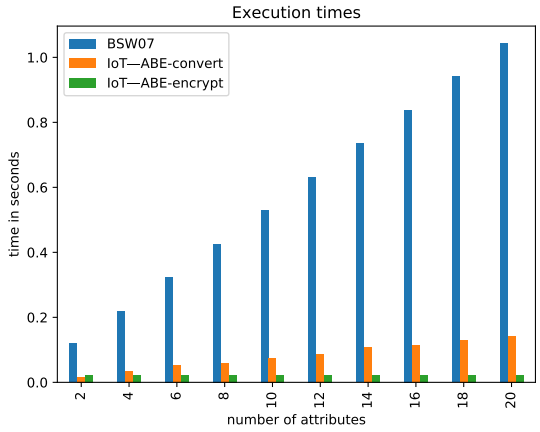

Fig. 3. Execution times of algorithms to create an ABE cipher-text in CP-ABE and IoT-ABE

machine, the sum of the execution time of the *Encrypt* and *Convert* algorithm of the IoT-ABE is lower, than the execution time of the original *Encrypt* algorithm.

The size of the created cipher-texts shows a similar behaviour as the execution time. In both implementations, the cipher-text size grows linear with the size of the access policy (number of attributes respectively). However, the size of the cipher-text grows in the modified *Encrypt* algorithm as well, from 469 Bytes with 2 attributes to 631 Bytes with 20 attributes. That is, because the output of the *Encrypt* algorithm already contains the access policy $\mathcal{T}$. As Fig. 4 shows, the cipher-text after the *Convert* algorithm is slightly bigger, than in the original CP-ABE scheme. This can be explained with the additional V embedded in the cipher-text. Because we assume the proxy has a high-bandwidth connection with the network storage, we do not think this poses a problem.

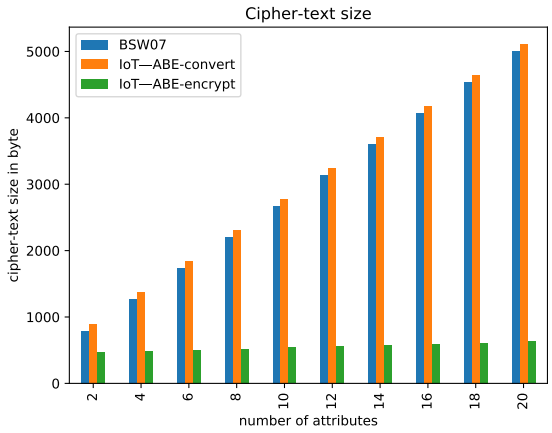

Fig. 4. Size of cipher-text in CP-ABE and IoT-ABE

VI. Key Revocation

There exists a conceptual problem with key revocation in ABE schemes. That is, there simply is no public key linked to

a specific user, that can be revoked. In ABE the "public key" is a set of attributes (or an access policy) describing one or many users in the system. In this paper, we define key revocation as the exclusion of one specific user from the system. A revoked user is not able to decrypt cipher-texts with his private key. We call the time between a user gets revoked and the revocation takes effect "revocation time". The key revocation mechanism in our modification works similar to the method employed by [13]. There, the attempt to download or decrypt a cipher-text resulted in a validation, to check if the user has been revoked. In our approach, a user, who wants to decrypt a cipher-text, needs to request additional information, namely the $\hat{D}$, from the AA. Upon this request, the AA can verify, if the user is revoked or not. In case the user is revoked, the AA replies with an error message.

By defining rules for the selection of v in the *Encrypt* algorithm, an operator of a sensor network with resource constraint IoT devices can configure the revocation mechanism. A newly chosen v in the *Encrypt* algorithm means, that a data users needs to obtain a new $\hat{D}$ by invoking the *Grant* algorithm at the AA. For example, he/she could define that a new random v has to be chosen only once a day, or v has to be renewed after it has been used a number of times. This increases the revocation time but reduces the communication and computation overhead to get $\hat{D}$. The sensor network provider can balance the trade-off between security (revocation time) and system performance.

VII. Conclusion

This paper presented a new approach to utilise ABE on resource constraint IoT devices. In this approach, most of the computations required to generate an ABE cipher-text are shifted to a new entity, a proxy. We showed, that the computational cost to create a cipher-text is constant with respect to the number of attributes used within the access policy and very low on the resource constraint IoT device compared to the original CP-ABE scheme. At the same time, the size of the cipher-text transmitted by the IoT device could be reduced. The proxy runs a *Convert* algorithm in order to convert a partial ABE cipher-text from an IoT device into a complete CP-ABE cipher-text. The proxy is never able to decrypt the cipher-text. Furthermore, our approach allows realising an instantaneous key revocation mechanism.

In future research, we will investigate a method to ensure the proxy uses the correct access policy in the *Convert* algorithm. At the moment, there is no way to ensure the proxy uses the access policy $\mathcal{T}$ provided as input through $\hat{E}$.

Acknowledgment

This work is part of the research project AgraSEC founded by the "Europäischen Fonds für regionale Entwicklung (EFRE)" (project number 85003218).

References

[1] Joseph A. Akinyele, Christina Garman, Ian Miers, Matthew W. Pagano, Michael Rushanan, Matthew Green, and Aviel D. Rubin. Charm: a framework for rapidly prototyping cryptosystems. *Journal of Cryptographic Engineering*, 3(2):111–128, 2013.

[2] Moreno Ambrosin, Mauro Conti, and Tooska Dargahi. On the feasibility of attribute-based encryption on smartphone devices. In *Proceedings of the 2015 Workshop on IoT challenges in Mobile and Industrial Systems*, pages 49–54. ACM, 2015.

[3] John Bethencourt, Amit Sahai, and Brent Waters. Ciphertext-policy attribute-based encryption. In *2007 IEEE symposium on security and privacy (SP'07)*, pages 321–334. IEEE, 2007.

[4] Dan Boneh and Matt Franklin. Identity-based encryption from the weil pairing. In *Annual International Cryptology Conference*, pages 213–229. Springer, 2001.

[5] Vipul Goyal, Omkant Pandey, Amit Sahai, and Brent Waters. Attribute-based encryption for fine-grained access control of encrypted data. In *Proceedings of the 13th ACM conference on Computer and communications security*, pages 89–98. Acm, 2006.

[6] Matthew Green, Susan Hohenberger, and Brent Waters. Outsourcing the decryption of abe ciphertexts. In *USENIX Security Symposium*, volume 2011, 2011.

[7] Sonia Jahid, Prateek Mittal, and Nikita Borisov. Easier: Encryption-based access control in social networks with efficient revocation. In *Proceedings of the 6th ACM Symposium on Information, Computer and Communications Security*, pages 411–415. ACM, 2011.

[8] Nouha Oualha and Kim Thuat Nguyen. Lightweight attribute-based encryption for the internet of things. In *Computer Communication and Networks (ICCCN), 2016 25th International Conference on*, pages 1–6. IEEE, 2016.

[9] Ronald L Rivest, Adi Shamir, and Leonard Adleman. A method for obtaining digital signatures and public-key cryptosystems. *Communications of the ACM*, 21(2):120–126, 1978.

[10] Amit Sahai and Brent Waters. Fuzzy identity-based encryption. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pages 457–473. Springer, 2005.

[11] Adi Shamir. Identity-based cryptosystems and signature schemes. In *Workshop on the Theory and Application of Cryptographic Techniques*, pages 47–53. Springer, 1984.

[12] Lyes Touati, Yacine Challal, and Abdelmadjid Bouabdallah. C-cp-abe: Cooperative ciphertext policy attribute-based encryption for the internet of things. In *Advanced Networking Distributed Systems and Applications (INDS), 2014 International Conference on*, pages 64–69. IEEE, 2014.

[13] Zhiqian Xu and Keith M Martin. Dynamic user revocation and key refreshing for attribute-based encryption in cloud storage. In *Trust, Security and Privacy in Computing and Communications (TrustCom), 2012 IEEE 11th International Conference on*, pages 844–849. IEEE, 2012.