CHIST-ERA

Use-IT

User empowerment for SEcurity and privacy in Internet of Things

Evaluation results Use-Cases

Deliverable number: D4.1

Version 1.0

Funded by the Future and Emerging Technologies (FET) CHIST-ERA programme of the European Union.

Project Acronym:	USEIT
Project Full Title:	User empowerment for SEcurity and privacy in Internet of Things
Call:	2015
Grant Number:	20CH21_167531
Project URL:	http://useit.eu.org

Editor:	CEA, UMU
Deliverable nature:	Report (R)
Dissemination level:	Public (PU)
Delivery Date:	August 2018
Authors:	Nouha Oualha, CEA Salvador Perez, UMU Antonio Skarmeta, UMU

Abstract

Provide a short description, highlight the topics considered in the deliverable.

The abstract must be self-contained, without abbreviations, footnotes, or references. It should be a microcosm of the full deliverable.

The abstract must be between 150-250 words. Be sure that you adhere to these limits.

The abstract must be written as one paragraph, and should not contain displayed mathematical equations or tabular material.

The abstract should include three or four different keywords or phrases, as this will help readers to find it. It is important to avoid over-repetition of such phrases as this can result in a page being rejected by search engines.

Ensure that your abstract reads well and is grammatically correct.

Contents

List of Figures

List of Tables

Executive Summary

This deliverable D4.1 evaluates the results of the tests specified in D3.2. The deliverable includes data gathered from the testing operations performed at the component and integrated system levels, and according to the test cases described in D3.2. The system validation is performed to ensure that the requirements identified in D1.1 are fulfilled. The deliverable includes also a formal security analysis of the proposed solutions.

List of Acronyms

ABE	Attribute-Based Encryption
C-ITS	Cooperative Intelligent Transport System
CP-ABE	Ciphertext-Policy Attribute-Based Encryption
CPU	Central Processing Unit
IDS	Intrusion Detection System

1. Introduction

The USEIT project develops data solutions that provide security and privacy for the IoT and empower users with the control over their data. In parallel to the implementation process, the project conducts an evaluation of such solutions implemented in different use case settings. The deliverable D4.1 presents the results of the evaluation process: first, the individual components of the proposed solutions are evaluated independently to ensure that these components work as specified and verify the security and privacy requirements identified in D1.1, then, the interfaces and the interactions between the components integrated together as specified in D3.1 are evaluated, and finally the project platform is evaluated in use case settings and validated with respect to the functional and non-functional final requirements identified in the final version of D1.1.

The deliverable is organized in two chapters associated with the identified two use cases:

- The Cooperative Intelligent Transport System (C-ITS) use case
- The smart objects use case

For each use case, the evaluation process will be performed at the component, integration, and system levels, followed by a validation of the project platform.

2. C-ITS use case evaluation results

2.1. Component test results

Testing is performed at low level considering only indivudal technical solutions.

2.1.1. Formal validation

2.1.2. Performance results

2.2. Integration testing results

Integration testing of interfaces between components are performed to ensure that they are compatible.

2.3. System testing results

Testing is realized over the entire system according to the test cases.

2.4. Validation

Evaluation at the end of system development is performed to ensure that the requirements identified in WP1 are fulfilled.

3. Smart objects use case evaluation results

To demonstrate the applicability of the proposed cryptographic solutions, a second use case, a smart building scenario, has been identified in WP2 and the associated requirements have been described. This chapter is dedicated to present the evaluation results of the project solutions with respect to the smart building use case.

3.1. Component test results

This section presents the results of component level testing considering only individual technical solutions.

3.1.1. Formal validation

3.1.2. Performance results

In this sub-section, performance evaluation of the implemented solutions are described, only focusing on individual components evaluated separately. The component evaluation considers two frameworks: a basic framework comprising smart objects collecting data and an extended framework involving Intrusion Detection System (IDS) probes in the network. In both frameworks, different performance metrics targeting smart objects consumption are evaluated.

3.1.2.1. Basic framework

This section is focused on demonstrating the advantages of the SymCpAbe approach in terms of performance. Towards this end, we compare our scheme with the direct application of CP-ABE to protect large amounts of data. Note that, in current CP-ABE schemes and implementations, each piece of data is protected by using a *one time symmetric key*, which is in turn encrypted with CP-ABE. Thus, each ciphertext includes both the encrypted data and the corresponding Ciphertext-Policy Attribute-Based Encryption (CP-ABE) encrypted symmetric key. Consequently, the distribution of the symmetric key is not required. Accordingly, we have considered this CP-ABE approach, since it is widely adopted in current works, such as [1, 2]. On the other hand, it should be pointed out that, under our SymCpAbe scheme, each symmetric key ($SYMK$) has associated a lifetime ($SYMK_{lifetime}$). Therefore, the corresponding *CP-ABE Delegator* will protect incoming data as long as this symmetric key is not expired; in case of key's expiry, a new key must be established by performing Phase 1 again. This way, if the $SYMK$ is obtained by an attacker, it will only be able to recover the data encrypted with such specific key. Furthermore, note that the $SYMK_{lifetime}$ is based on the number of published events in order to delimit the amount of data that could be accessed in an unauthorized way, regardless of the *CP-ABE Delegator* publication rate.

According to the entities, functionality and hardware/software components described in D3.2 for the smart objects basic framework, we now show a performance evaluation of our proposal by considering different practical aspects, in particular, the runtime, memory consumption and the number of attributes of the CP-ABE access policy.

Data Event Publication Performance

This stage comprises the set of steps and operations required to protect and send the data to the *Event Storage Service* (in this case, a publish/subscribe broker). Specifically, it covers Phases 1-3 in the case of SymCpAbe, while for CP-ABE, each data are protected by using such encryption scheme. It should be noted that SymCpAbe results have been obtained by considering "$SYMK_{lifetime}$ = 1 event". Therefore, this can be considered as the *"worst case"* for our approach since Phases 1 and 2 must be performed every time a data is received by the *CP-ABE Delegator*. Thus, Figure 3.1 shows the memory consumed by the *CP-ABE Delegator* to publish a new encrypted data event by using both approaches. As shown, while the memory consumption increases according to the number of attributes in the access policy for CP-ABE, it remains constant under our approach since the CP-ABE encryption operation is delegated to the *CP-ABE Assistant* (Phase 2). Similarly, Figure 3.2 shows the average runtime required for this stage. Thus, in case of CP-ABE, it increases linearly (from 1105 ms for 1 attribute to 6525 ms for a 10-attribute access policy). Note that, even for the SymCpAbe *"worst case"*, the required runtime grows very slowly (from 328 ms to 857 ms for 1 and a 10-attribute access policy, respectively). Furthermore, it should be pointed out that if "$SYMK_{lifetime} > 1$ event", the runtime would be decreased, since the most time-consuming phase (Phase 2) is only executed when $SYMK_{lifetime}$ expires.

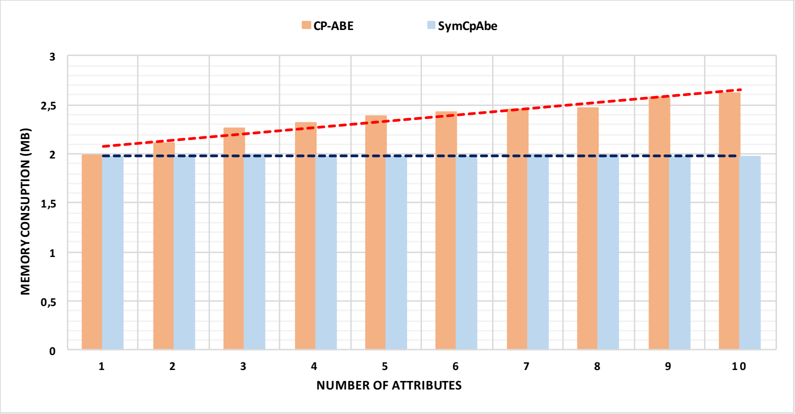

Figure 3.1: Memory consumption for CP-ABE and SymCpAbe by the *CP-ABE Delegator*

Data Event Retrieval Performance

This stage comprises the operations required by *Applications* to get the data from the *Event Storage Service* (a publish/subscribe broker) and decrypt them. In particular, it covers Phase 4 in the case of SymCpAbe, while for CP-ABE, the *Application* are responsible for decrypting each data by using such scheme. Thus, Figure 3.3 shows the memory consumption by considering CP-ABE and SymCpAbe approaches according to the number of attributes in the access policy. While in the case of CP-ABE the memory consumption remains constant, in the case of SymCpAbe, this value is slightly increased. Indeed, with the direct application of CP-ABE, the *Application* only needs to perform the CP-ABE decryption operation to get access the data of the event. In contrast,

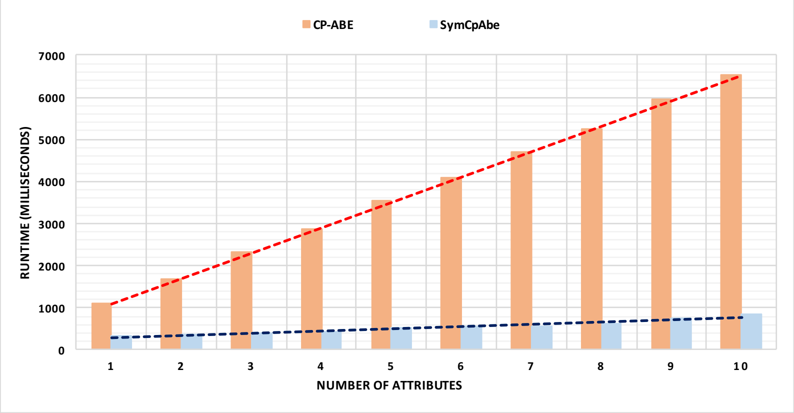

Figure 3.2: Runtime for CP-ABE and SymCpAbe by the *CP-ABE Delegator*

using our approach, it should firstly contact the *Key Storage Service* to get the $SYMK$ that was used to encrypt such data. Moreover, Figure 3.4 shows the runtime required by *Applications* to retrieve data with both approaches. It should be pointed out that we have considered different cases according to the value of $SYMK_{lifetime}$. As shown, only for the SymCpAbe *"worst case"*, the performance of the CP-ABE approach is better, since for that case, the *Application* should get a new $SYMK$ for each received event. Indeed, when the $SYMK_{lifetime}$ is increased, the performance of SymCpAbe is better than CP-ABE, as shown in such figure. This is because the most expensive operations (i.e., getting the $SYMK$ and decrypting it by using CP-ABE) are only required in case that a new $SYMK$ is used by the *CP-ABE Delegator* to protect the data.

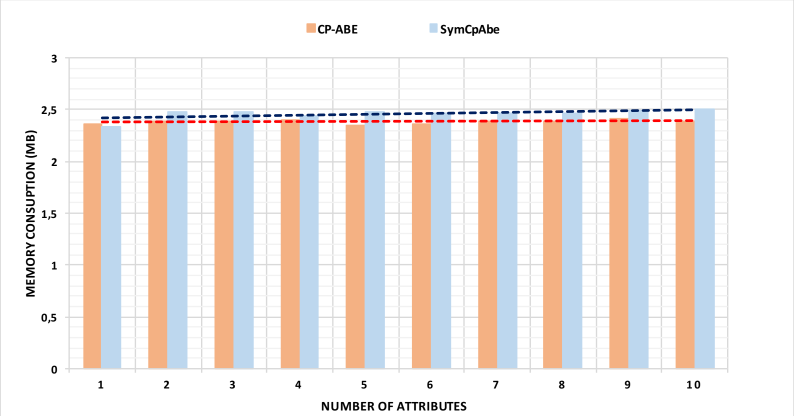

Figure 3.3: Memory consumption for CP-ABE and SymCpAbe by the *Application*

According to these results, it is demonstrated that our scheme represents a more efficient and scalable approach than the direct application of CP-ABE when both schemes are used on scenarios where large amounts of data need to be protected. However, it should be pointed out that the inclusion of additional components gives rise to further security aspects to be considered. Specifically, by using SymCpAbe, the *CP-ABE Assistant* has the keys that are used by *CP-ABE Delegators* to encrypt data. Consequently, it could access the data from the *Event Storage Service* in case they are required. This fact represents an inherent aspect to be considered since the users' privacy could

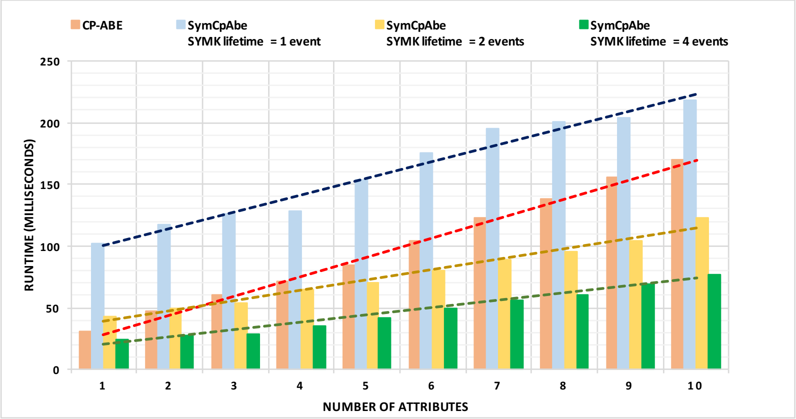

Figure 3.4: Runtime for CP-ABE and SymCpAbe by the *Application*

be threatened. In this sense, we have considered the *CP-ABE Assistant* as a semi-trusted service (i.e., honest but curious), so it does not confabulate with other entities to use such data with malicious intent. Additionally, this service could still be authorized by the *Key Storage Service* and the *Event Storage Service* to access both the encrypted keys and the encrypted data. In this sense, authorization models based on our access control approach based on capabilities [3] could be further integrated. In addition, in order to overcome the problems arising from the use of a single entity for the CP-ABE encryption (i.e., the *CP-ABE Assistant*), alternative approaches based on outsourcing CP-ABE operations could be applied, such as [4]. Specifically, in our case, the *CP-ABE Delegator* would be able to outsource the CP-ABE encryption of the $SYMK$ without the need to disclose such key itself. However, note that even in this situation, if cryptographic operations are outsourced to more powerful entities, network overhead could still involve a significant issue in certain scenarios.

3.1.2.2. Extended framework

The extended framework comprises a network composed of sensor devices that collect measurements. Before transmitting data, sensor devices encrypt their data using Attribute-Based Encryption (ABE) schemes. Two largely known existing ABE schemes are implemented: the Bethencourt et al.'s scheme [5] and the Waters'scheme [6]. These two schemes are extended using pre-computation techniques as described in [7] for the Bethencourt et al.'s scheme. The pre-computation technique allows to design lightweight variants of the two ABE schemes in terms of Central Processing Unit (CPU) usage in exchange for more memory space. The generator parameters of the technique are configured as $n = n_e = 256$ and varying $k = 1, 2, 3, 6$ (parameters should be large enough to avoid Birthday attacks).

Figure 3.5 and figure 3.6 show the execution time consumed by the encryption algorithm on the sensor device over the number of attributes in the access policy associated with the data for Bethencourt et al's and waters schemes and their variants using pre-computation techniques. Values are calculated as the average over different access policies. The figures show that the execution time increases, almost in a linear fashion, with the number of attributes in the access policy.

Figure 3.5 demonstrates that the pre-computation technique applied to the Bethencourt et al.'s

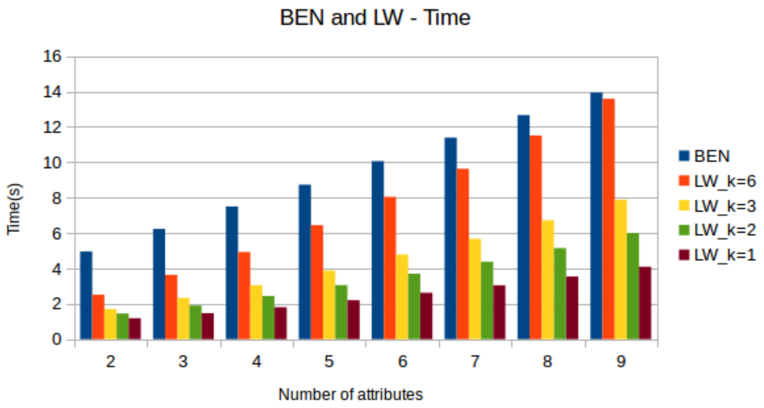

Figure 3.5: Time consumption of Benthencourt et al.'s scheme and Benthencourt et al.'s scheme with pre-computation technique

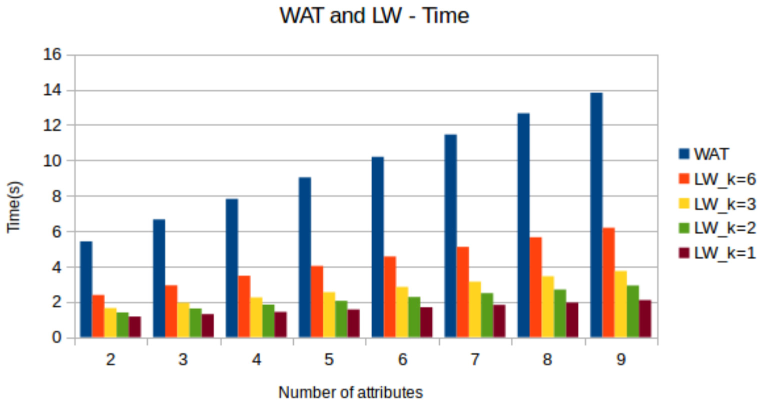

Figure 3.6: Energy consumption of Waters' scheme and Waters' with pre-computation technique scheme

scheme allows to reduce half of the execution time of the encryption algorithm, when the value of k is smaller than 3. For value of k is 6, the Waters's scheme is more efficient. The pre-computation technique is more interesting when applied to the Waters' scheme, since for a value of k of 6, the execurition time is decreased by half, as shown in figure 3.6.

Energy consumption is of a primary importance for battery-powered sensor devices. In order to estimate the energy consumption, we used tools provided by the Contiki OS [8]. The total energy consumption E is computed using the following formula:

$$E/V = I_m t_m + I_l t_l + I_t t_t + I_r t_r + \sum_i I_{c_i} t_{c_i}$$

where, $V = 3V$, $I_m = 0.6mA$, $I_l = 1.3\mu A$, $I_t = 24mA$, $I_r = 20mA$.

Figure 3.7 and figure 3.8 show the total energy consumption of the two implemented schemes over the number of attributes in the access policy. Values are calculated as the average over different access policies. The two figures demonstrate that the energy consumption increases, approximately linearly, with the number of attributes.

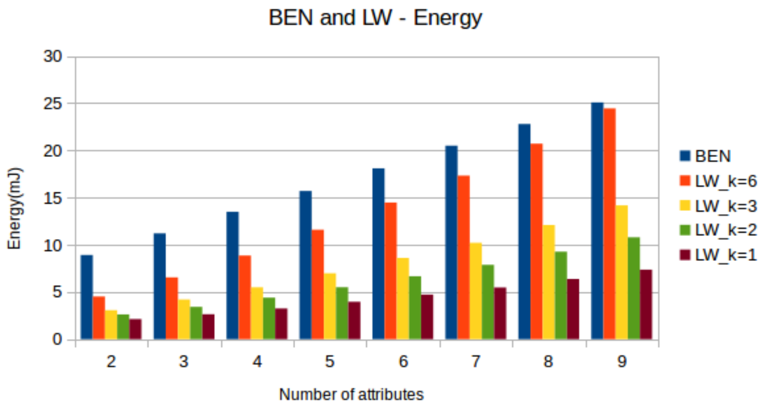

Figure 3.7: Energy consumption of Benthencourt et al.'s scheme and Benthencourt et al.'s scheme with pre-computation technique

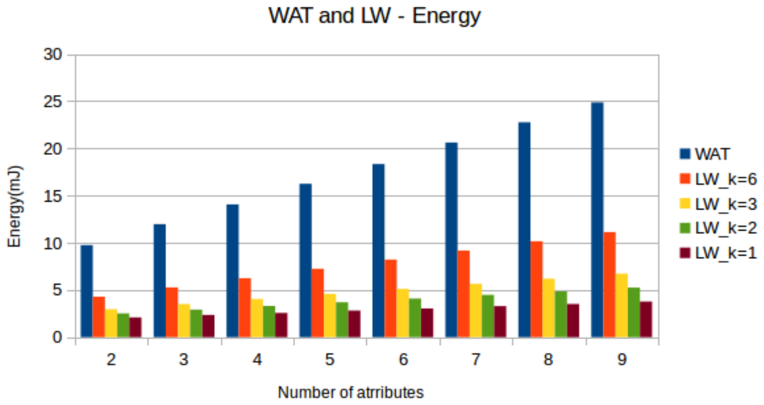

Figure 3.8: Energy consumption of Waters' scheme and Waters' scheme with pre-computation technique

Figure 3.7 shows that the Bethencourt et al.'s scheme outperforms when the value of k is larger than 3. With lower value of k, this situation is reversed. Figure 3.8 demonstrates that the pre-computation technique applied over the Water's scheme achieves around 60% of gain in terms of energy consumption. For instance, the energy consumption with pre-computation increases with around $1mJ$ when the access policy has one more attribute, while with the basic scheme, it increases with around $3mJ$. The gain achieved with the pre-computation technique increases exponentially with the number of attributes.

The runtime memory and storage management is important to assure the stability of resource-constrained devices. As shown in TABLE **??**, the code size of the encryption algorithm for each of the four implemented schemes is around 225 to 233 kB, which is suitable for the Remote revision B platform with 500 kB flash memory.

The main drawback of the pre-computing technique is that it requires, in addition to the algorithm code, the storage of a sufficient number of pre-computed values. The storage space required may

Table 3.1: Notations for the energy consumption formula

V	The supply voltage
I_m	The current draw of the microprocessor when running
t_m	The time in which the microprocessor has been running
I_l and t_l	The current draw and the time of the microprocessor in low power mode
I_t and t_t	The current draw and the time of the communication
I_r and t_r	The current draw and time of the communication device in receive mode
I_{c_i} and t_{c_i}	The current draw and time of other components such as sensors and LEDs

Table 3.2: Code size of the encryption algorithm of the implemented schemes

Bethencourt et al.‘s (Bytes)	Waters‘s (Bytes)	Bethencourt et al.‘s with pre-computation (Bytes)	Waters‘s with pre-computation (Bytes)
225879	228679	233718	232235

reach 1MB. All tuples of pre-computed values are stored in the flash memory using a micro SD card, since the RE-Mote Zolertia revision B is equipped with an external storage.

3.2. Integration testing results

Integration testing of interfaces between components are performed to ensure that they are compatible.

3.3. System testing results

Testing is realized over the entire system according to the test cases.

3.4. Validation

Evaluation at the end of system development is performed to ensure that the functional and non-functional requirements identified in WP1 are fulfilled.

Table 3.3: Functional and non-functional requirements associated with smart objects use case

ID	Type	Priority	Description	System validation
SOREQ1	NFREQ	MUST	Producer's data integrity must be supported	-
SOREQ2	NFREQ	MUST	Producer's data confidentiality must be supported	-
SOREQ3	NFREQ	MUST	Obtaining cryptographic keys and credentials requires strong authentication	-
SOREQ4	NFREQ	MUST	Secure key storage	-
SOREQ5	NFREQ	SHOULD	Secure signature (verification) outsourcing	-
SOREQ6	NFREQ	SHOULD	Secure encryption/decryption outsourcing	-
SOREQ7	NFREQ	MAY	Distributed cryptographic outsourcing	-
SOREQ8	NFREQ	MUST	Collusion resistance	-
SOREQ9	NFREQ	SHOULD	Use of scalable cryptographic algorithms beyond the use of symmetric-key cryptography approaches	-
SOREQ10	NFREQ	SHOULD	Use of privacy-preserving signature algorithms	-
SOREQ11	NFREQ	SHOULD	Use of flexible encryption algorithms	-
SOREQ12	NFREQ	MUST	Policy-based approaches for defining security and privacy preferences	-
SOREQ14	NFREQ	MAY	Use of transport layer security mechanisms	-
SOREQ15	NFREQ	MUST	Access to platform's data or services will be protected	-
SOREQ16	FREQ	MUST	Access to platform's data or services could be done in a privacy-preserving way	-
SOREQ17	FREQ	MUST	The platform must allow publish/subscribe interactions	-
SOREQ18	FREQ	SHOULD	Information and semantics models should be supported by the platform	-
SOREQ19	FREQ	MAY	The platform must provide storage facilities	-
SOREQ20	FREQ	MAY	The platform must provide analytic facilities	-

4. Conclusions

This deliverable describes the evaluation results of the testbeds associated with the two selected use cases in the project: the C-ITS and the smart objects use cases. Following the agile methodology, the evaluation is performed while realizing the implementation of the proposed solutions associated with the two use cases. In each iteration of the evaluation, the project solutions are evaluated by checking whether they satisfy new identified requirements. The analysis of the evaluation results will allow to improve the solutions specified in D3.2 and the architecture of the system proposed in D3.1.

Bibliography

[1] X. Yao, Z. Chen, and Y. Tian, “A lightweight attribute-based encryption scheme for the internet of things,” *Future Generation Computer Systems*, vol. 49, pp. 104–112, 2015.

[2] M. Singh, M. Rajan, V. Shivraj, and P. Balamuralidhar, “Secure mqtt for internet of things (iot),” in *Communication Systems and Network Technologies (CSNT), 2015 Fifth International Conference on*. IEEE, 2015, pp. 746–751.

[3] J. L. Hernandez-Ramos, M. P. Pawlowski, A. J. Jara, A. F. Skarmeta, and L. Ladid, “Toward a lightweight authentication and authorization framework for smart objects,” *IEEE Journal on Selected Areas in Communications*, vol. 33, no. 4, pp. 690–702, 2015.

[4] J. Li, X. Huang, J. Li, X. Chen, and Y. Xiang, “Securely outsourcing attribute-based encryption with checkability,” *IEEE Transactions on Parallel and Distributed Systems*, vol. 25, no. 8, pp. 2201–2210, 2014.

[5] J. Bethencourt, A. Sahai, and B. Waters, “Ciphertext-policy attribute-based encryption,” in *2007 IEEE Symposium on Security and Privacy (SP '07)*. IEEE, 2007.

[6] B. Waters, “Ciphertext-policy attribute-based encryption: An expressive, efficient, and provably secure realization,” Cryptology ePrint Archive, Report 2008/290, 2008, https://eprint.iacr.org/2008/290. [Online]. Available: https://eprint.iacr.org/2008/290/20101220:203013

[7] N. Oualha and K. T. Nguyen, “Lightweight attribute-based encryption for the internet of things,” in *2016 25th International Conference on Computer Communication and Networks (ICCCN)*. IEEE, aug 2016.

[8] A. Dunkels, F. Osterlind, N. Tsiftes, and Z. He, “Software-based on-line energy estimation for sensor nodes,” in *Proceedings of the 4th workshop on Embedded networked sensors - EmNets '07*. ACM Press, 2007.

A. Appendix

A.1. List of Abstracts of Papers on the Work Reported Herein

1.