Efficient Access-Control in the IIoT through Attribute-Based Encryption with Outsourced Decryption

Dominik Ziegler[1], Alexander Marsalek[2], Bernd Prünster[3] and Josef Sabongui[2]
[1]*Know Center GmbH, Austria*
[2]*Institute for Applied Information Processing and Communications (IAIK), Graz University of Technology, Austria*
[3]*Secure Information Technology Center – Austria (A-SIT), Austria*
dominik.ziegler@tugraz.at, alexander.marsalek@iaik.tugraz,at, bernd.pruenster@a-sit.at, josef.sabongui@student.tugraz.at

Keywords: ABE, IIoT, Fine-Grained Access Control, Attribute-Based Encryption, OAuth, eID

Abstract: We present a new architectural design to leverage Attribute-Based Encryption (ABE) in the Industrial Internet of Things (IIoT). The general idea of our approach is to automatically issue and revoke attributes based on already established identity management systems. Our design enables organisations to rely on arbitrary identity and access management solutions across different security domain boundaries. We, furthermore, tackle privacy concerns typically associated with outsourcing sensitive data to the cloud. To demonstrate the feasibility and versatility of our approach, we evaluate our design by integrating both OAuth and the Austrian eID. Besides, we present performance data. The evaluation results clearly show that our proposed design suits the requirements imposed by the IIoT well.

1 Introduction

Attribute-Based Encryption (ABE) is a cryptographic way to achieve fine-grained access control on encrypted data. There are two main flavours of ABE: Key-Policy Attribute-Based Encryption (KP-ABE) (Goyal, Pandey, Sahai, & Waters, 2006) and Ciphertext-Policy Attribute-Based Encryption (CP-ABE) (Bethencourt, Sahai, & Waters, 2007). KP-ABE schemes embed access control policies into a party's secret key. CP-ABE schemes incorporate access-structures in the ciphertext.

Problem. While ABE promises fine-grained access control, one of the major challenges remaining is how to authenticate and authorise users in heterogeneous environments. Indeed, ABE does not provide a standardised way to do so. The emergence of the Industrial Internet of Things (IIoT) has also raised the question about the performance of established ABE schemes. A primary concern is that conventional ABE systems rely on high computational power which cannot possibly be provided in the IIoT. They also do not account for already deployed legacy clients. Despite questions raised about the practicability of ABE in the IIoT, little is known how service providers can integrate existing solutions. An approach, which allows defining fine-grained access rights in a context separated from the service provider and which accounts for legacy and lightweight clients is still missing.

Approach. We evaluate how ABE can efficiently be used in IIoT environments. Consequently, we tackle this issue from three angles. First, we evaluate how service providers can enforce existing access control policies. Second, we demonstrate how ABE can be applied in heterogeneous environments. Finally, we evaluate our approach under realistic constraints for the targeted scenario to prove its feasibility.

Contribution. Our contributions are as follows:

1. We present an architectural design to combine established Identity and Access Management (IAM) solutions with ABE. Our approach supports arbitrary protocols and allows managing policies in a context, separated from the service provider.
2. To demonstrate the practicality of our approach, we provide a software implementation. We integrate OAuth and the Austrian eID, to demonstrate compatibility with two distinct IAM schemes.
3. We present performance data and discuss the overhead of our implementation.

Outline. We present our results in five parts. First, we describe the architecture of our approach in Section 2. We present the solution of our system and illustrates how it can be used in practice. Next, we evaluate our approach in Section 3. Section 4 discusses previous research in this area and highlights the contribution of this work. Finally, the conclusion provides both a summary and a critical review of the findings.

2 Architecture

We, first, give a high-level overview of the architecture, consisting of two separate tenants. We introduce all entities and showcase the workflow. Next, we explain the necessary building blocks. We rely on the concepts proposed by Hur and Noh (2011), Green, Hohenberger, and Waters (2011) and Lin, Hong, and Sun (2017) as the base ABE system. Their works address several typical challenges of IIoT systems, including instant attribute revocation, backward and forward secrecy, and performance. In general, however, any ABE system which offers similar properties is suitable for the proposed architecture.

2.1 Entities

Data Owner (DO). A Data Owner produces (encrypted) information for entities in the system to consume. DOs are typically lightweight components.
Client (CL). A Client is a party trying to access a resource through a front-end device.
Identity Provider (IdP). The Identity Provider is responsible for authenticating users. It issues attributes and allows administrators to add or revoke identities.
Resource Provider (RP). The Resource Provider acts as a transparent gateway between an enterprise and a service provider tenant. It handles decryption of ciphertexts and is responsible for attribute management.
Key Authority (KA). The Key Authority is the sole authority to grant attributes and the associated ABE keys.
Re-Encryption Server (RS). The Re-Encryption Server (1) transforms ciphertexts from DOs to ensure attribute revocation and (2) stores them.
Decryption Server (DS). The Decryption Server performs most parts of expensive operations necessary to (partially) decrypt a ciphertext.

2.2 Workflow

Figure 1 displays the steps in the system. First, a DO encrypts some data under an access structure. Next, it ① uploads the resulting ciphertext to the RS. Clients can now ② request access to this resource. They, first ③ authenticate with the IdP. The RP can then ④ retrieve the necessary identity attributes for a user. It creates an attribute attestation and ⑤ sends the digitally signed claim to the KA. Afterwards, the RP ⑥ fetches the requested ciphertext, from the RS. In a conventional ABE system, the RP would now have to perform an expensive decryption operation. In our system, we delegate this task to a powerful proxy, provided by some service provider. The RP, thus, ⑦ offloads the partial decryption operation to the DS. Finally, the DS ⑧ obtains the decryption keys from the KA and the RS to partially decrypt the ciphertext. Afterwards, the DS ⑨ sends the now transformed ciphertext to the RP. In the last step, the RP extracts the plaintext by a simple exponentiation and ⑩ sends it to the client.

2.3 System Setup

We setup the ABE system, in three steps. First, we define the public base system parameters, based on some security parameter λ. We select some bilinear group of prime order and define generators for the groups. Next, we initialise the system and generate random elements for each attribute in the system. Furthermore, we generate public parameters for the KA and the RS.

2.4 Key Creation & Update

Generating keys is a two stepped process. First, the KA and the RS generate initial keys. To do so, they calculate a shared secret over a Multi-Party Computation (MPC) protocol. Both actors then compute their initial keys based on the calculated secret and some random element. In the second step, the KA can issue private keys for the clients. We refer to this process as key update. Analogous to the initial key agreement, the KA and RS first agree on their respective private keys using their initial keys. Next, the KA, the RS and the RP carry out a MPC protocol for each client CL. The resulting private key contains all attributes the client holds. Relying on this two stepped process has two advantages. First, decryption keys are split between multiple parties. Hence, no (malicious) actor in the system alone can decrypt a ciphertext. Second, whenever a key needs to be revoked, the KA and the RS only need to run the key update protocol to issue a new key.

2.5 Encryption

To encrypt a plaintext, a DO first generates some random element m. Together with a random Initialisation Vector (IV), the element will act as input to a hash function from which the payload key is derived. The DO then encrypts the plaintext with the payload key. Next, the DO can encrypt the random element under the public parameters of the RS. It, furthermore, generates a random secret s and calculates a share matrix from the access structure. In the last step, in encrypts the required attributes under the public key of the KA. The DO now generates the initial ciphertext. It consisting of the payload, the random element m, the share matrix and the attributes. Finally, the DO sends it to the RS.

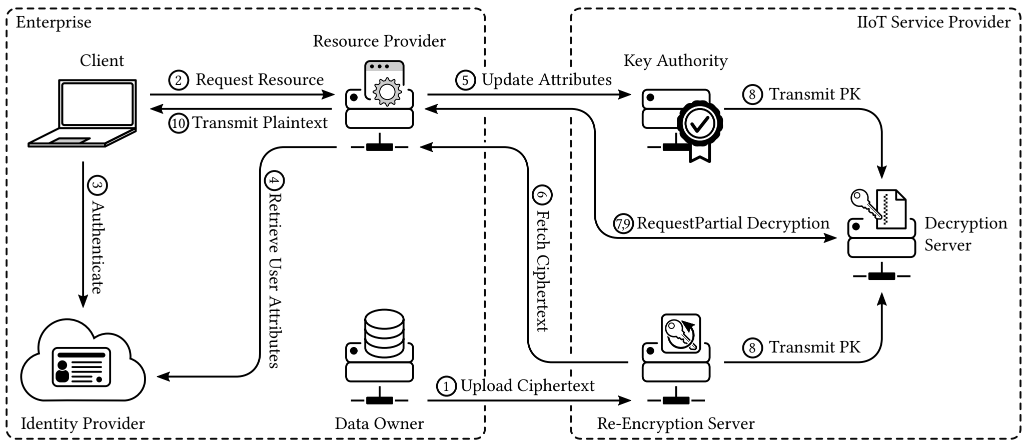


Figure 1: System Architecture: Two separate tenants for the service provider and enterprise.

2.6 Re-Encryption

Once the RS receives an initial ciphertext, it transforms it into a suitable format for revocation and partial decryption. To do so, it derives a unique element for each CL, based on its unique identifying bits. The algorithm then generates random attribute group keys for each encrypted attribute. To recover these attribute group keys in a later step, a ciphertext header Hdr is necessary. It allows the DS to solve equations based on its entries. Finally, all attribute ciphertext parts are re-encrypted with the respective key. Subsequently, the Hdr is appended to the final ciphertext CT.

2.7 Decryption

To decrypt a ciphertext, two steps are necessary. First, the RP sends the ciphertext, requested by some client, to the DS. The DS, then, retrieves the key parts from the KA and the RS. Using the header Hdr of the ciphertext, it then attempts to recover the attribute group keys. If successful, the DS recovers the ciphertext secret s by solving a system of linear equations, given the CLs authorised attributes. Finally, it can perform the partial decryption of the random element m and returns it to the RP. In the second step, the RP can recover the payload key by a simple exponentiation and decrypt the payload.

2.8 Authentication

To authenticate users, the RP first retrieves attribute information from some IdP. It, then, creates an assertion and sends the claims to the KA. We rely on JSON Web Tokens (JWTs) (Jones et al., 2015) and Elliptic Curve Digital Signature Algorithm (ECDSA) to securely transfer these attribute claims between the RP and the KA. The KA can thus verify that the asserted claims are legitimate.

2.9 Attribute-Update & Revocation

Updating user attributes is as simple as adding them to attribute groups. To revoke attributes, users are simply removed from affected groups. Changing attribute groups, however, entails that all ciphertexts which contain the affected groups need to be re-encrypted. This means that new group keys are chosen to re-encrypt the corresponding ciphertext parts, with the keys being protected by generating new ciphertext headers (see Section 2.4, Section 2.6). Still, re-encryption is costly. As a result, we apply a lazy re-encryption mechanism. We set a re-encryption flag on all affected ciphertexts instead of re-encrypting them. One advantage of this approach is that updating attributes can be achieved in an instant. Ciphertexts only get re-encrypted on first access or when system load is low.

2.10 Security Requirements

We now analyse the basic security properties of our design. We do not consider attacks on the implementation itself and assume the correctness of all involved cryptographic primitives. We assume that parties are honest but curious. Entities are curious about sensitive data. The communication among parties is assumed to be secure and authentic. We further assume that data in the enterprise tenant belong to the organisation and not to the individual user. We define the security requirements and behaviour of each entity and their attack vectors through a theoretical analysis.

Curious Entities: Assume one of the entities in the service provider tenant is curious about data they receive. Then it should not be possible to infer any information about the data itself. Indeed, in our protocol DOs always encrypt plaintext data before sending them to another entity. The used Advanced Encryption Standard (AES) key is encrypted under a CP-ABE policy. No entity in the service-provider can decrypt the ciphertext alone, since the keys are split between KA, RS and RP. Hence, no entity in the service-provider tenant can infer any information about the encrypted data. Only the designated client, RP respectively, can decrypt data if the client has the correct attributes.

Malicous Entities: Assume an entity in the system has been compromised. Then an adversary should not be able to decrypt arbitrary ciphertexts. Let us look at the properties of each of the two tenants separately. Malicious entities in the service provider tenant can decrypt initial ciphertexts if they obtain both private keys of KA and RS. Since the MPC protocol ensures that no party has both keys, a single malicious entity cannot decrypt arbitrary ciphertexts. Adversaries in the enterprise tenant can decrypt ciphertexts if they obtain a suitable secret key. The RP is the single authority to update user attributes and to retrieve client secret keys. As a result, the RP must not be compromised.

Revoked User: Assume a user in the system needs to be revoked. Then a revoked user should not be able to infer any information about encrypted ciphertexts.

Once an attribute is revoked, affected attribute groups will be updated and new attribute group keys will be calculated. As a result, affected secret keys cannot be used to decrypt previously obtained ciphertexts. Replaying old messages also does not work, as the DS needs to retrieve the newly calculated public keys of KA and RS.

3 Experimental Results

We provide a Java and Kotlin based software implementation to establish the practicality of the solution. We evaluate the solution and discuss our findings.

3.1 Evaluation

We tested the implementation regarding two metrics: Computation overhead, and storage and communication impact. We encrypted a random ciphertext as described in Section 2.5. We repeated each test 100 times on a single core on an Intel(R) Xeon(R) CPU E5-2699 v4 @ 2.20GHz. We relied on the *IAIK Provider for the Java^{TM} Cryptography Extension (IAIK-JCE)*[1] and the *IAIK ECCelerate^{TM}* for bilinear pairings and encryption operations. We used the *FRESCO* (Alexandra Institute, 2019) library for the MPC computation. We tested our implementation with policies containing up to $n = 100$ attributes. We use an asymmetric bilinear map $e : \mathbb{G}_1 \times \mathbb{G}_2 \rightarrow \mathbb{G}_T$ with security parameter $\lambda = 256$. The bit-size of $\mathbb{G}_1$ is 265-bit, whereas the bit-size of $\mathbb{G}_2$ is 520-bit. The resulting group $\mathbb{G}_T$ is 3072-bit. This level is equal to near-term security (at least ten years) as defined by ECRYPT-CSA (ECRYPT – CSA, 2018) and NIST (Barker, 2016).

[1]https://jce.iaik.tugraz.at

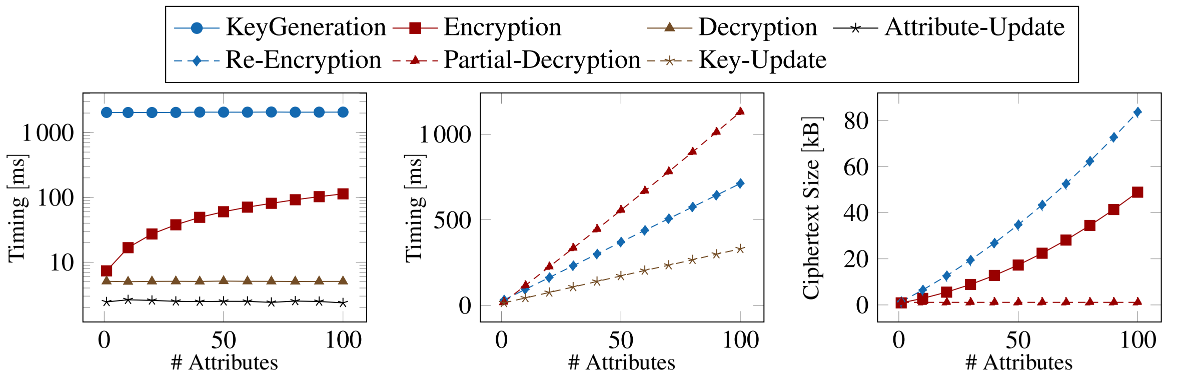
3.1.1 Computation Overhead

Figure 2a, shows the execution time for the operations in the enterprise tenant. We are using a logarithmic scale with a base of 10. The data are quite revealing in several ways. First, we can see that the execution time of the encryption operations grows linearly with the number of attributes. Second, encryption takes just a few tenths of a second for a small number of attributes and only 100 ms for up to 100 attributes. Third, the graph confirms our hypothesis that decryption is constant due to it being a simple ElGamal operation. Interestingly, the graph further reveals that generating a key is constant for the enterprise tenant. We attribute this to the fact that generating the key is merely executing the MPC protocol. The key agreement between KA and RS does not affect the performance on the client-side. From the data, we see that updating attribute groups is constant and independent from the number of attributes in the enterprise tenant. This can be attributed to the fact that all subsequent operations (e.g. re-encryption and key-update) are performed in the service provider tenant.

In turn, Figure 2b, shows the timing of all operations in the service provider tenant. The data confirm our hypothesis that partial-decryption is among the most expensive operations in the system. The second most expensive operation is the re-encryption operation. Lastly, the graph shows the raw computation time for the key-update operation.

3.1.2 Storage and Communication Overhead

Figure 2c shows the ciphertext size for encryption, re-encryption and partial decryption operations. As expected, the size of the ciphertext increases with the number of attributes for encryption and re-encryption operations. It is independent of the plaintext data, though, as only the constant size AES key is encrypted. Our tests reveal that the initial ciphertext after the encryption operation consumes approximately up to 50 kB when enforcing a policy with 100 attributes. The re-encrypted ciphertext consumes up to 80 kB. Our tests further reveal that the partially decrypted ciphertext is constant sized and consumes not more than 1 kB in our implementation. This can be attributed to the fact that the partially decrypted ciphertext represents only a single element in $\mathbb{G}_T$.



(a) Operations in Enterprise Tenant (b) Operations in Service Provider Tenant (c) Ciphertext Size for attributes

Figure 2: Execution time and ciphertext size of involved ABE operations for security parameter $\lambda = 256$.

3.2 Integrating established IdPs

We showcase the steps necessary, to integrate both, Open Authorization (OAuth) (Hardt, 2012) and the Austrian eID into the proposed system. This process allows us to draw conclusions about the interoperability with industrial systems. Indeed, the Austrian eID features a tight security concept with a strong focus on user privacy. As a result, the lessons learned can also be applied to evaluate the requirements of dedicated corporate eID solutions. Our use case assumes an enterprise tenant with a predefined OAuth or Austrian eID provider exists. Only the RP needs to be extended, to integrate the proposed system into this environment. We identify three steps. First, the RP needs to be modified to support transforming attestations issued by the IdPs. This procedure consists merely of implementing the necessary protocols for the IdPs and transforming the claims, as described in Section 2.8. Second, to support attribute revocation, a similar process is applied. The RP only needs to validate tokens obtained from the IdP. If a token is expired, the associated user is removed from the assigned attribute-groups, and issued keys are invalidated. If an attribute is removed from the scope, the RP creates an attestation with the remaining attributes. Affected CLs are not able to decrypt ciphertexts any more. Thus, existing revocation strategies can be used. Third, matching access control policies need to be created to account for attributes issued by the IdP. Conveniently, attribute-based access control policies are flexible enough to represent arbitrary scopes. All that is necessary is creating an access structure combining all necessary attributes.

3.3 Discussion

Our initial experiments show that our system can indeed be used in industrial environments to provide fine-grained access control. All expensive operations are executed in some service provider tenant, which benefits from on-demand resource allocation and scalability. Operations in the enterprise tenant are either constant or scale linearly with the number of attributes. The communication overhead for our solution is at most 80 kB for a ciphertext with 100 attributes. Admittedly, while these figures do not seem as large overhead, given today's infrastructure, it can put a significant strain on the infrastructure in a highly dynamic environment. As a result, a separate evaluation for those environments needs to be considered. By integrating existing IAM solutions, we have furthermore shown that our approach is valid and achieves granular access control across different security domain boundaries. The dynamic process of our solutions even allows us to model arbitrary access control policies or to mandate specific IdPs. Our analysis also shows that the proposed system achieves its goal to protect sensitive data. Data are always encrypted before leaving the enterprise tenant. Decryption keys are split between RP, KA and RS, Hence, the service provider cannot learn the plaintext since no involved party has access to all parts. In certain scenarios, the RP can, however, be considered as a weak spot or a single point of failure. In fact, the RP can easily be eliminated, at the cost of backwards compatibility, to increase the security of the system further. All operations performed by the RP must then be performed by the client or the IdP.

4 Related-Work

We summarise the major contributions related to ABE in cloud computing and discuss their relationship to our work. Wang, Liu, and Wu (2010) tackle the security and privacy issues related to outsourcing sensitive data to Cloud Service Providers (CSPs). The authors present a hierarchical ABE scheme where domain masters can administer users on behalf of a root master. Our architecture differs from their scheme by providing a full architecture for heterogeneous environments and does not rely on require an attribute history list for revocation. M. Li, Yu, Zheng, Ren, and Lou (2013) proposed a design which leverages ABE to provide secure data sharing of personal health records. The scope of their work does not, however, include resource-constrained devices or existing infrastructure. J. Li, Zhang, Chen, and Xiang (2018) present a lightweight approach for secure fine-grained data sharing. Similar to our construction, the majority of computations can be offloaded to more powerful devices. Their scheme does not support direct attribute revocation, as our architecture does. Sepehri and Trombetta (2017) present an inner product-based, attribute-based proxy re-encryption scheme. Like our design, their scheme is suitable for computationally constrained scenarios. The authors do not discuss attribute revocation or authentication across multiple security domains. Michalas (2019) proposed a new protocol which combines ABE and Symmetric Searchable Encryption (SSE) to overcome current weaknesses of cloud storage: Missing user revocation and overall efficiency. The authors separate revocation from ABE by using Intel's Software Guard Extensions (SGX) to deploy a revocation authority in a Trusted Execution Environment (TEE). Notwithstanding, their protocol does not concentrate on the performance of ABE itself. Hence, decryption and thus, multiple bilinear pairing operations are still costly from a users perspective. Summarising, in this paper, we bridge the gap between heterogeneous environments and fine-grained access control. We improve the state-of-the-art by proposing a new design to incorporate ABE while keeping security guarantees for data owners.

5 Conclusion

By presenting an architectural design based on Attribute-Based Encryption (ABE) with outsourced decryption, we demonstrated how secure resource sharing can be achieved across different security domain boundaries. Our approach differs from existing solutions by providing an industry-first approach, focusing on resource-constrained devices and support for arbitrary Identity Providers (IdPs). Existing workflows, as well as user revocation, remain unchanged. The conducted evaluation proves that the proposed design is practical. The most apparent finding is that the proposed system only introduces a nominal overhead for organisations.

The current system does not account for user privacy or accountability of actions. Indeed, the issue of privacy and accountability in these professional environments is an intriguing one. We consider the privacy of users as one of the most significant improvements to the system, in the future. Furthermore, we will concentrate on improving the overall performance of the system.

References

Goyal, V., Pandey, O., Sahai, A., & Waters, B. (2006). Attribute-Based Encryption for Fine-grained Access Control of Encrypted Data. In *Proceedings of the 13th ACM Conference on Computer and Communications Security* (pp. 89–98).

Bethencourt, J., Sahai, A., & Waters, B. (2007). Ciphertext-Policy Attribute-Based Encryption. In *2007 IEEE Symposium on Security and Privacy (SP '07)* (pp. 321–334).

Hur, J., & Noh, D. K. (2011). Attribute-Based Access Control with Efficient Revocation in Data Outsourcing Systems. *IEEE Transactions on Parallel and Distributed Systems*, *22*(7), 1214–1221.

Green, M., Hohenberger, S., & Waters, B. (2011). Outsourcing the Decryption of ABE Ciphertexts. In *Proceedings of the 20th USENIX Conference on Security* (p. 34). Berkeley, CA, USA: USENIX Association.

Lin, G., Hong, H., & Sun, Z. (2017). A Collaborative Key Management Protocol in Ciphertext Policy Attribute-Based Encryption for Cloud Data Sharing. *IEEE Access*, *5*, 9464–9475.

Jones, M., Bradley, J., & Sakimura, N. (2015). *JSON Web Token (JWT)* (RFC No. 7519). RFC Editor. RFC Editor.

Alexandra Institute. (2019). FRESCO - A FRamework for Efficient Secure COmputation.

ECRYPT – CSA. (2018). *D5.4 Algorithms, Key Size and Protocols Report (2018).* H2020-ICT-2014 – Project 645421.

Barker, E. (2016). *Recommendation for Key Management Part 1: General*. National Institute of Standards and Technology.

Hardt, D. (2012). *The OAuth 2.0 Authorization Framework*. RFC Editor. RFC Editor.

Wang, G., Liu, Q., & Wu, J. (2010). Hierarchical Attribute-based Encryption for Fine-grained Access Control in Cloud Storage Services. In *Proceedings of the 17th ACM Conference on Computer and Communications Security* (pp. 735–737).

Li, M., Yu, S., Zheng, Y., Ren, K., & Lou, W. (2013). Scalable and Secure Sharing of Personal Health Records in Cloud Computing Using Attribute-Based Encryption. *IEEE Transactions on Parallel and Distributed Systems*, *24*(1), 131–143.

Li, J., Zhang, Y., Chen, X., & Xiang, Y. (2018). Secure attribute-based data sharing for resource-limited users in cloud computing. *Computers & Security*, *72*, 1–12.

Sepehri, M., & Trombetta, A. (2017). Secure and Efficient Data Sharing with Atribute-based Proxy Re-encryption Scheme. In *Proceedings of the 12th International Conference on Availability, Reliability and Security* (63:1–63:6).

Michalas, A. (2019). The Lord of the Shares: Combining Attribute-based Encryption and Searchable Encryption for Flexible Data Sharing. In *Proceedings of the 34th ACM/SIGAPP Symposium on Applied Computing* (pp. 146–155).