electronics

MDPI

Article

Fine-Grained Access Control with User Revocation in Smart Manufacturing

Ernesto Gómez-Marín [1,*], Davide Martintoni [2], Valerio Senni [2], Encarnación Castillo [3] and Luis Parrilla [3]

[1] Infineon Technologies AG, 85579 Neubiberg, Germany
[2] Applied Research & Technology, Collins Aerospace, 00185 Rome, Italy; davide.martintoni@collins.com (D.M.); valerio.senni@collins.com (V.S.)
[3] Departamento de Electrónica y Tecnología de Computadores, Universidad de Granada, 18071 Granada, Spain; encas@ugr.es (E.C.); luis@ugr.es (L.P.)
* Correspondence: ernesto.gomezmarin@infineon.com or ernesto.gomezmarin@gmail.com

Abstract: Collaborative manufacturing is a key enabler of Industry 4.0 that requires secure data sharing among multiple parties. However, intercompany data-sharing raises important privacy and security concerns, particularly given intellectual property and business-sensitive information collected by many devices. In this paper, we propose a solution that combines four technologies to address these challenges: Attribute-Based Encryption for data access control, blockchain for data integrity and non-repudiation, Hardware Security Modules for authenticity, and the Interplanetary File System for data scalability. We also use OpenID for dynamic client identification and propose a new method for user revocation in Attribute-Based Encryption. Our evaluation shows that the solution can scale up to 2,000,000 clients while maintaining all security guarantees.

Keywords: Industrial Internet of Things; access control; blockchain; attribute-based encryption; revocation; data-sharing; Industry 4.0

Citation: Gomez-Marin, E.; Martintoni, D.; Senni, V.; Castillo, E.; Parrilla, L. Fine-Grained Access Control with User Revocation in Smart Manufacturing. *Electronics* **2023**, *12*, 2843. https://doi.org/10.3390/electronics12132843

Academic Editors: Georgios Kavallieratos, Georgios Spathoulas and Marios Anagnostopoulos

Received: 19 May 2023
Revised: 20 June 2023
Accepted: 23 June 2023
Published: 27 June 2023

1. Introduction

The third industrial revolution was a paradigm shift due to the inclusion of computers and programmable elements in the industry. However, the fourth industrial revolution brings a new manufacturing change—this time not by the inclusion of new machines, but by the advancement of information and communication of the machines and the autonomous manufacturing [1,2]. One of the fundamental elements of this revolution is precisely the machine-to-machine interaction, or rather, thing-to-thing through the internet, with the aim of sharing information and self-organizing to face the changes in the environment. This is called the Internet of Things (IoT) [3].

IoT is currently being used in the industry for a large variety of use cases. For this reason it has received its own name—Industrial IoT (IIoT). Some of the applications are: Predictive maintenance by gathering information from the machinery, such as temperature or vibration, optimizing the supply chain with asset tracking by sending the asset location in real-time, inventory management by monitory the inventory in real-time, quality control by detecting defects in the product or failures in the manufacturing process, etc. [4]. Furthermore, IIoT is growing rapidly. A recent study by The Business Research Company claims a growth from USD 209 billion in 2022 to USD 252 billion in 2023, and they also forecast that IoT in the manufacturing market will grow to more than USD 400 billion in 2027 [5].

For all those use-cases, data has to be shared not only with machines in the same factory level but with individuals from different levels and even with other stakeholders, through vertical and horizontal data flow. The potential value of sharing data between different partners is estimated to be more than USD 100 billion [6]. This Collaborative Manufacturing (CM) brings the ability to react to the customers' needs and requirements and provide

a significant advantage [2]. Thereby, data sharing between different stakeholders of the product life cycle, regardless of location, is a must for Industry 4.0 [7,8]. Here IIoT plays a vital role as data producers for production flow, quality control, product traceability, supply chain, and enterprise decision management [7]. However, IIoT inherited from IoT poses a big challenge: overcoming the security concerns raised by the need to actually connect IIoT the Internet, in order to make it part of a larger and distributed network. This is due to the great security risk involved in this leap, i.e., 83% of the IoT devices used or manufactured by large companies experienced a cyber attack in 2018 and 79% of the manufacturing organizations admitted to being victims of a cyberattack in their IIoT the same year [9].

IoT nodes can be remotely attacked by exploiting their vulnerabilities [10,11] and on the other hand, outgoing messages could be modified to attack the infrastructure or the messages could simply be read in the flow [12]. However, this is only part of the problem, as data normally needs to be stored to be eventually shared. However, data storage for IIoT presents two big challenges. The reliability of cloud storage is not trusted (1), it can fail and lose data accidentally, e.g., such as the failures of Amazon Web Services, Microsoft Azure, Alibaba Cloud and Google Cloud [13–16], fog storage can intentionally modify the data to affect the user operation, and local storage is not an option due to the huge quantity of data generated by IIoT [7]. The other big challenge is ensuring the data confidentiality (2) [7,12]. A common method to protect data confidentiality is by encryption in the form of cipher-text, and then forwarding the decryption key to the data readers. The problem with this mechanism is the complexity of managing accurate access control rules in order to provide least-privilege access to sensitive data and proprietary knowledge. Moreover, agents requiring data access may belong to different entities, have different levels of authorization, and have complex enroll/leave dynamics. Therefore data sharing with fine-grained authorization is a relevant challenge in IIoT [7].

There are many works in the state-of-the-art trying to solve this problem. Still, none of them can provide a completely secure solution, e.g., Ref. [17] ensured fine-grained authorization, but did not provide a method to protect the integrity of the data. On the other hand, Ref. [18] did protect the integrity of the data but did not propose any mechanisms to revocate the users. Therefore, they can ultimately not be applied in their current status.

This paper proposes a complete secure mechanism to collect and share data from IIoT devices with multiple readers with fine-grained access authorization. The proposed solution is tested and evaluated on a smart manufacturing use-case scenario identified in the context of the H2020 COLLABS project [19]. The contributions of the current paper can be summarized as follows:

- Maximization of data availability by using decentralized databases based on Inter-Planetary File System (IPFS) [20].
- Data integrity though blockchain-based mechanisms.
- Strong identity and authenticity of IIoT through hardware-based encryption.
- Confidentiality in-transit and at-rest through end-to-end encryption.
- Fine-grained data access control by defining specific attributes and policies per datum through the use of Attribute-Based Encryption (ABE).
- Dynamic enrollment based on the OpenID standard [21].
- A novel reader revocation mechanism, beyond the capabilities of current research on fine-grained data access control.

The paper is structured as follows: Section 2 presents the industrial use case and its requirements; Section 3 reviews the related work; Section 4 presents the core technical concepts used in the proposed solution; Section 5 introduces the proposed solution from a high-level point of view; Section 6 explains in detail the design of the technical innovation achieved by this paper; Section 7 provides insights on the prototype implementation, discusses the evaluation results, and how the solution fits the use-case; Section 8 explains the findings of our work by comparing it with the state-of-the-art, highlights the limitations

and discusses how they could be addressed as future work; Finally, Section 9 presents the conclusions.

2. Use Case and Requirements

The scenario described in this paper is a real-world scenario provided by Collins, one of the contributing organizations of this paper. In the context of safety-critical products, such as in aerospace systems, collecting and sharing data throughout the manufacturing lifecycle is of the utmost importance: the information produced is not only used for coordination of the manufacturing supply chain, but it is also used for quality controls or to prove standard adherence providing the final item with a safety certification that is strictly required to its usage. Of course, in the same context, the information required for a proficient CM might include important data containing sensitive information or intellectual property that must be protected.

Figure 1 represents a high-level architecture of the data flows. Notice that the architecture is divided into four different zones, each of which is mapped to the corresponding Purdue Enterprise Reference Architecture level, also known as Purdue Level (PL) [22]. These levels are used to relate a specific architecture to key enterprise layers. Specifically, PL0 represents the physical process; PL1 includes devices that are directly interacting with the physical processes (i.e., sensing); PL2 describes the control systems that are monitoring and controlling the physical processes (i.e., SCADA); PL3 represents the manufacturing operations that manage production flows (i.e., manufacturing execution/operations); PL4, on the other hand, consists of IT networks, where the enterprise runs its business functions (i.e., engineering functions).

The following presents an introduction of the zones depicted in Figure 1:

- Shop Floor: this layer represents the inner-most layer of the enterprise, where the actual manufacturing activities are performed by industrial machines. This corresponds to PL0, PL1, and PL2.
- Manufacturing zone: this layer hosts all the industrial functionalities that directly manage the shop floor such as SCADA servers and other industrial controls. This corresponds to PL2, PL3, and PL4.
- Enterprise: this zone includes all the high level services of the company that are used to coordinate, manage, and operate the manufacturing operations such as the engineering center and the IT infrastructure. This corresponds to PL4.
- Public Internet: this zone represents all the services collaborating with the production of the final product that are not inside the company itself, for example, cloud services and supply chain partners. This layer is not mapped to any PL since it is physically and logically outside of the company.

The data flows shown in the architecture can be divided into two types: the black arrows identify data flows produced by equipment deployed on the shop floor, and the green arrows identify the data flow consumed by entities localized in any of the four layers. Notice that different types of devices are involved in the data collection phase, for example, modern industrial machines and sensors, which can be categorized as IIoT. On top of that, a typical shop floor usually hosts legacy industrial devices that might not be able to interface directly with the data-sharing system and therefore can use an industrial gateway as a trusted proxy to send their data to the data storage. All these devices gathering data during the manufacturing phases (for example manufacturing parameters such as production temperature, time, or precision) send this data to a central location (here identified as data storage) to be shared with the desired data consumers (hereafter called clients) afterward. Clients can analyze this information for performance improvement or quality assurance. On the other hand, the green lines identify data consumers: these data flows identify a set of possible clients that need access to the data collected throughout the manufacturing lifecycle. The clients can be found at all levels of the architecture and might include employees of the factory (i.e., production manager, SCADA manager or engineers), automated data analysis services (i.e., quality and performance

analysis tools or computing cloud services) or external partners (i.e., suppliers, customers, or certification authority). The shared data contains critical proprietary knowledge of the manufacturers and other stakeholders that shall be shared only with the specified client. Therefore, all the aforementioned data consumers must get access only to the minimum amount of data required to perform their duty (least knowledge principle [23]). However, obtaining fine-grained authorization in this scenario is complicated because managing diverse readers with different level of authorization and requirements can become highly intricate. In addition, thousands of IIoT devices generating data require ensuring that they are correctly configured with their policies and authorizations, which further complicates the authorization process. Additionally, clients and IIoT devices will dynamically join and leave the network requiring revocations, enrollments, and authorization updates, making the scenario a challenging undertaking.

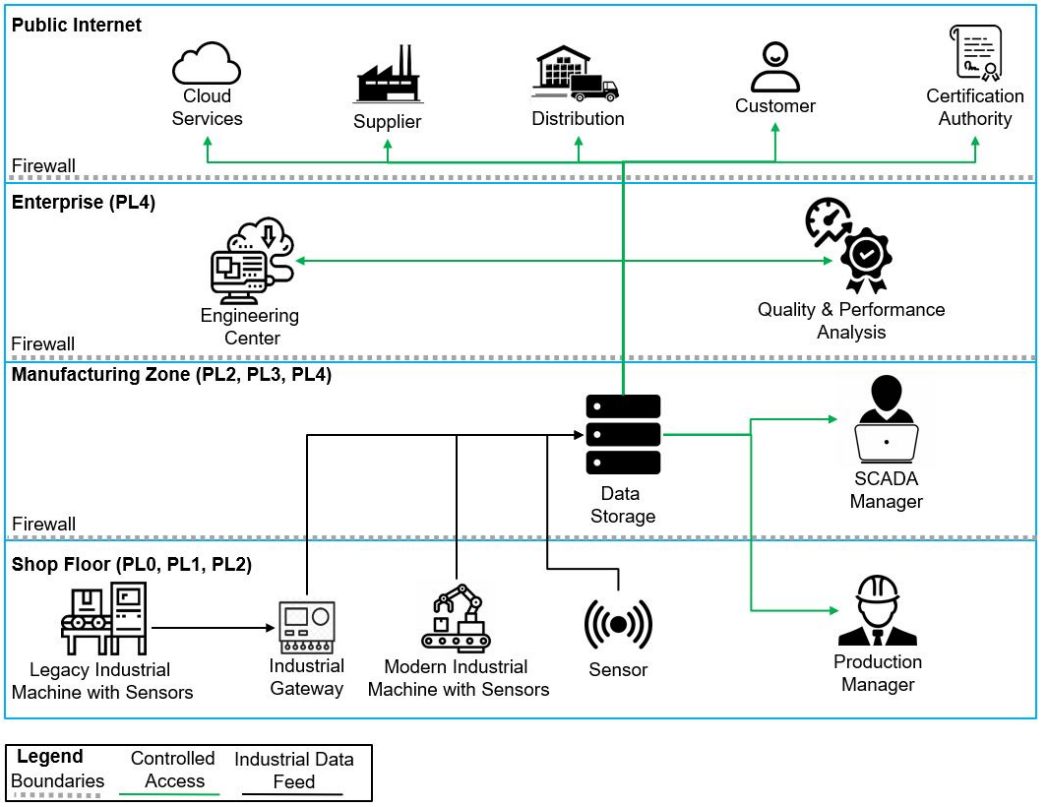

Figure 1. High-level scheme of factory scenario.

With the collaboration of a forum of industry experts involved in the COLLABS project [19], a list of high-level requirements for the CM dataflows identified [24] in the reference architecture has been redacted:

RQ1: data confidentiality shall be protected while in transit and while at rest (end-to-end encryption).
RQ2: clients shall have access only to the minimum amount of information required to perform their duty (least knowledge principle).
RQ3: clients shall have access to the information only within a specific time-frame (time limited access).
RQ4: information managed by the system shall have integrity guarantees to be used as proof in a certification process.
RQ5: the solution architecture should support scalability and multi-site integration to be applicable in manufacturing scenarios with globally distributed manufacturing sites.
RQ6: data in the system shall guarantee authenticity to provide confidence in its origin.

RQ7: data in the system shall have non-repudiation guarantees to establish trust and accountability between the parties involved.

3. Related Work

This section presents the solutions in the state-of-the-art to provide confidentiality to large quantities of data with fine-grained authorization while ensuring the integrity of the data from generation to consumption.

Some solutions for access control tend to rely on the cloud services, which may compromise data security and privacy due to the reliance on a single cloud service provider with unrestricted access to all data. Moreover, the recent failures of major cloud providers such as Amazon Web Services, Microsoft Azure, Alibaba Cloud, and Google Cloud [13–16] have highlighted the risks involved in relying solely on cloud services. To address this issue, classic cryptography techniques (such as symmetric and asymmetric cryptography) can reduce the required trust in cloud service providers. However, providing fine-grained access using these techniques remains a challenge.

A possible approach is to define a user list per file [25], but this requires the encrypter to know the data consumers beforehand and perform a specific encryption per data consumer (readers), which is not applicable to a distributed IoT-based environment with a high number of readers. A solution to this problem is Attribute-Based Encryption (ABE) [26], and its later variation Ciphertext-policy ABE (CP-ABE) [27] where a central entity provides attributes to the readers, and the encryptor can easily define the combinations of attributes needed to decrypt its data. However, this solution does not offer efficient user revocation [28] or does not provide data integrity and authenticity guarantees.

There are multiple solutions trying to address the revocation problem. Historically, the proposed solutions were theoretical works proposing new algorithms or modifications of the original, such as Refs. [29,30], which allow a dynamic membership. However, the first one requires trustful servers to avoid collaboration of servers and revoked users, which could lead to a collision attack. In addition, the server could modify the files during the re-encryption without being noticed. In the second work, after every revocation and key update, all the old messages are not reachable anymore with the new key, so a newly joined user cannot decrypt the previously published ciphertexts, i.e., they do not provide forward security [31]. Other theoretical solutions to this problem were presented in Refs. [32,33], but they suffered from revocation collision attacks, (i.e., a revoked user can collaborate with a non-revoked user to recover the decryption capability of the revoked user accessing data that they could not get separately).

Revocation in ABE is still a challenging issue. On top of that, the lack of reliability, traceability, and availability in data storage services has also caught the attention of researchers as it affects the overall security of the infrastructure. To avoid relying on a third party to store the immutable proofs, decentralization stands out as a possible solution. The authors of Ref. [18] propose to store encrypted data (ciphertext) in the InterPlanetary File System (IPFS) [20]. IPFS is a decentralized data storage service that can handle large volumes of data, and it uses the ciphertext's hash as a locator, similar to a Uniform Resource Locator (URL) on the web. In addition, they store the ciphertext's hash on a blockchain. This research achieves reliability and transparency using blockchain, while benefiting from the availability and scalability of IPFS, and finally taking advantage of the fine-grained authorization of ABE. The approach proposed in Ref. [18] is similar to the scheme proposed in this paper but does not provide revocation nor allow multiple writers. In Ref. [34], blockchain is used to deliver attributes to consumers transparently and decentralized. In addition, the blockchain performs a pre-decryption of the ciphertext to relieve IoT devices of part of the decryption process while keeping confidentiality. To avoid users abusing the decryption services, they implement a user credibility incentive scheme. In Ref. [35], the authors create a Privacy-preserving Blockchain Energy Trading Scheme to manage energy production and sales transparently using blockchain and keeping the user

privacy through using ABE. However, none of these three papers propose any revocation mechanisms for their systems.

Indeed, some works exploit the synergies of ABE and blockchain while claiming to have revocation mechanisms. In Ref. [36], an infrastructure is created to share data from IoT nodes with fine-grain authorization. They propose to store the data encrypted with a symmetric encryption key in the cloud. The corresponding symmetric encryption key is encrypted with CP-ABE, which is called a header, and is stored in a smart contract. A client has to satisfy the policy defined in the smart contract in order to download the header. In this system, by modifying the smart contract, it is possible to modify the data access policy and thus to revoke users.

Another approach, described in Ref. [37], is an infrastructure that manages patient data shared among various hospitals and medical entities. This work aims to give patients the control over the access to their data. In this system, patients publish their data access attributes or policies in a permissionless blockchain. The patient or hospital then publishes a header in a permissioned blockchain, which only accepts storing the header if it is encrypted according to the guidelines defined in the permissionless blockchain. Using a "transaction consumption" mechanism, the patient can modify access to the header stored in the permissioned blockchain, and data consumers will not receive the data from the permission blockchain despite having an authorized access structure in the ABE encryption.

In Ref. [38], data is encrypted using a secret key K, which is generated through the secret keys K1 and K2. All consumers have access to K1, but K2 is encrypted with CP-ABE on the blockchain. All the nodes in the blockchain have the master key of CP-ABE and the attributes of all the consumers but lack K1, so they cannot access data even if they have K2. Eventually, when the consumer requests access to K2 in the blockchain, the blockchain nodes verify their attributes and, if valid, generates their decryption key to decrypt K2 and return it to the consumer. The consumer with K1 and K2 can then generate K and decrypt the message.

In Ref. [17] the authors use a Java-based blockchain to perform all the vital functions of ABE: generation of master key and the public key of ABE, and generation of the secret keys of each user. They obtain revocation by re-encrypting the ciphertexts using a group-based attribute access policy instead of the whole access policy.

The two last papers [17,38] fail in a common element; they rely on the correct behavior of the individual blockchain nodes. A permissionless or permissioned blockchain is reliable for data integrity because the failure or dishonesty of one node is corrected by the others, so it can only be attacked with a "51% attack", i.e., more than half of blockchains miners agree to attack the system, which is very unlikely. However, a blockchain is not that reliable for confidentiality and privacy [39] since the failure or the dishonesty of only one of the blockchain nodes can lead to a sensitive data leak (e.g., a individual blockchain node can provide unauthorized data from the ledger to the user). This single-node misbehavior could lead to data leaks in both works. Additionally, Refs. [17,36,38] do not have any mechanism to verify the integrity of the data received by the consumer. Finally, Refs. [36–38] do not have complete revocation mechanisms because they base their revocation on a policy update without giving further details, which means modifying the attributes and their relation in the crypto data. However, revocation is produced by an unexpected event that affects a user, not attributes, and therefore, it must be addressed to individual users [40,41]. A policy update of the crypto data will very likely affect non-revoked users, therefore, it is needed to explain how to revoke only to the specified users without affecting non-revoked users.

Finally, Guangsheng Yu et al. [42] proposes a solution that addresses various problems related to storing encrypted data in blockchain. To ensure data integrity, they store the encrypted data directly in blockchain, using scalability solutions such as OmniLedger. They also introduce a new type of blockchain, called the Redactable Key Chain, which is based on Chameleon Hash and allows modification of hashes and blocks where they store the headers of the encrypted messages. This allows the Redactable Key Chain to modify blocks through a consensus mechanism, and then modifying the headers of the

encrypted data. Their revocation mechanism is indirect revocation, which means updating the keys of all users except those of the revoked users. To perform the update of all keys in a scalable way, they use a concept called "updating factor". When a non-revoked user receives an "updating factor", they can update their keys. In addition, they provide another "updating factor" to the servers so they can update the headers without access to them. This solution has the following problems: it requires updating all the headers every time there is a revocation to guarantee forward security, i.e., new users have access to old data if they satisfy the policies [31]; it has no revocation collision resistance, i.e., non-revoked user Alice can share the Updating factor with revoked user Bob to gain access to data that only Bob could access if he were not revoked. Additionally, the concept of the "updating factor" is not well explained or supported by existing research.

As can be observed, many works are seeking a secure implementation of ABE and have found blockchain to provide a great synergy. However it is a big challenge to achieve revocation with collision resistance revocation, blockchain-based integrity protection, and without storing sensitive information in a blockchain. Nevertheless, these four features are necessary to apply ABE to real-world scenarios. This paper presents a probable secure solution for all these challenges. The proposed approach builds on Ref. [18], and improves it by providing a self-designed revocation mechanism, a dynamic user registry, and strong authenticity guarantees for multiples data senders, representing a significant step forward in the application of ABE in real-world scenarios. The reader can find a comparison of the shortcomings of the state-of-the-art and our work in Section 8.

4. Background

Our work relies on three existing technologies to achieve the security features that we claim: CP-ABE, blockchain, and Hardware Security Modules.

4.1. CP-ABE Functions

This work uses the CP-ABE solution proposed by Brent Waters in Ref. [43], Appendix A. In this construction, the set of attributes U is unlimited and the public parameters are constant. In CP-ABE, the readers have decryption keys with a set of attributes S, and the writers can encrypt a raw message (M) with public keys, defining a list of attributes combined with *and*/*or* logic functions called policy P, e.g., *Area51 and worker or manager*.

The attribute set has to satisfy the policy in order to decrypt the ciphertext (CT):

- Setup(): The setup function is executed by the trusted entity. The output is the Master Secret Key (MSK) and the ABE Public Key (ABE-PK). Once executed, these two elements remain constant.
- KeyGen(MSK, S): The key generation function is also executed by the trusted entity. It takes the MSK and S. The output is an ABE Decryption Key DK with attributes S.
- Encrypt(ABE-PK,P,M): The encryption function can be executed by any entity. It takes as input ABE-PK, and M. The output is CT.
- Y(P,S) The satisfaction function $Y()$, checks if the attributes in the set S satisfy the policy P; if yes return 1, if not, return 0.
- Decrypt(CT,DK): This function takes an input a CT and a DK. The output is M if and only if $Y(P, S)$, where P is the policy defined during the encryption of CT and S is the set of attributes established in the generation of DK. In the rest of the cases, the output is $\perp$.

Different policies can be easily merged with *and* and *or* gates.

4.2. Blockchain

The blockchain as a distributed ledger was first proposed by Satoshi Nakamoto in 2008 [44]. Since then, this technology has gained significant attention and undergone numerous design variations. In 2014, Ethereum [45] introduced a new concept to blockchain with the creation of public and transparent script—the smart contracts. Nowadays the term blockchain identifies a set of distributed ledgers technologies where data can change only through a specific process. Data stored in blockchains is guaranteed to be immutable,

meaning it can never be altered or deleted. This is accomplished through the use of cryptographic hashes, digital signatures, and distributed consensus protocols. For example, adding information in a blockchain requires a consensus among the network's nodes. This consensus is achieved if the network participants verify the new information and agree to add them to the blockchain. In addition, to avoid the use of certificates in some blockchains, the use of "addresses" instead IDs is widespread. The address is a unique bit array computed from the public key with a collision resistance function. In this way, given a public key, it is possible to bind it with its assigned address without using a certificate. Blockchains can be public ledgers that everyone can join, or they can enforce authentication and authorization systems to enable participation as peers or clients. The latter type of blockchain are referred to as permissioned blockchains. The present work is developed using the permissioned blockchain Hyperledger Fabric [46].

4.3. Hardware Security Modules

Hardware Security Modules (HSM) are dedicated hardware implemented according to security-by-design principles and offering core security services and cryptographic operations. The elemental functions of an HSM are true random number generator used to generate private keys, secure data storage for the private credentials, and secure implementation of the standard cryptographic operations such as AES, ECC, or RSA. The HSMs are prepared to resit hardware-based attacks and Side-channel attacks [47]. Therefore, verifying the signature performed by a private key generated by an HSM ensures that the signature was also signed using this same HSM. These are only the more basic functionalities; however, more sophisticated HSMs such as those following the Trusted Platform Module 2.0 standard (TPM2.0) [48] offer more complex tools so that they can be used to remotely attest the software of an IoT device (Remote Attestation) or to specify policies before using the private keys, e.g., delivering a pin or proving adherence to a software status. Thereby, HSM can offer big reliability over the IoT devices and their operations, and are a perfect fit to answer IoT security requirements [49].

5. Proposed Scheme

This section presents the main functionalities of the proposed system from a high-level point of view. Here follows a list of the entities involved in the system:

- *Sensor:* this label represents all the devices in the Shop Floor that collect essential data and send them to the data storage.
- *Sensor Owner (SO):* is the entity in charge of the correct operation of the sensors.
- *Client:* as explained in Section 2, a client is a data consumer that needs access to specific sensor data. They can be located in any of the four layers.
- *ID PRovider (IDP):* the entity that identifies the clients and their data requirements. Through the use of Open ID this entity can identify the *Client* and its attributes to *SO*.
- *IPFS message broker:* it is a decentralized message broker that can handle high volumes of data. Because the hash of the data is used as data locator, the integrity of the data is as trusted as the integrity of the hash.

Using the entities from the list, a simple sequence of actions is presented to showcase the system's capability. In the following example, a set of sensors is initiated with crypto material (Private/Public Keys, certificates, identifiers, etc.) and the related access policy. In this context, two users (Alice and Bob) will be used to demonstrate the possible operations of the systems, including encryption, decryption, and user revocation. The following lists explain the corresponding step number in Figure 2.

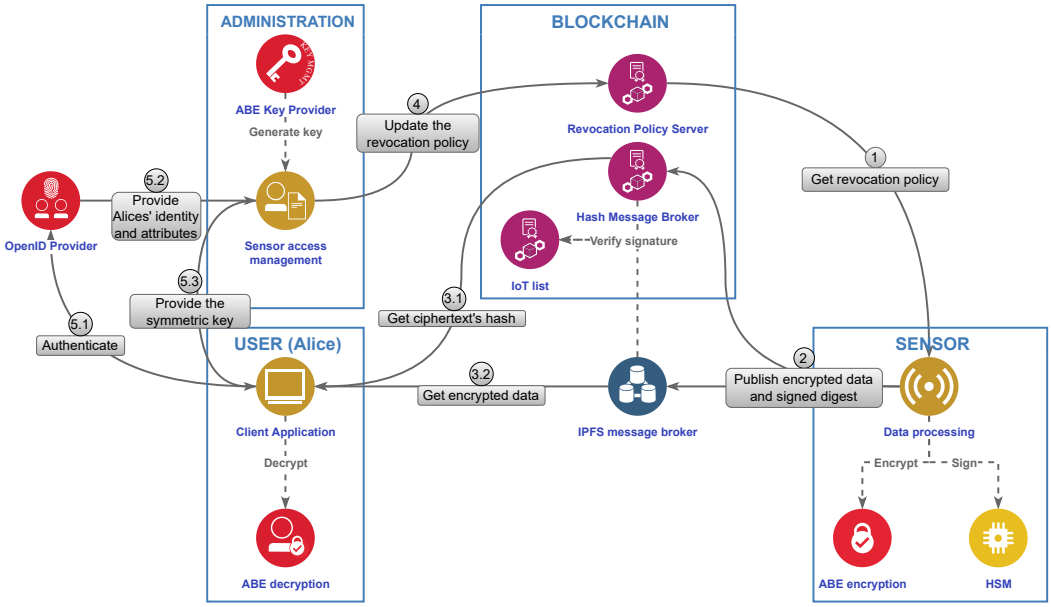

Figure 2. High-level scheme of the proposed solution.

0. **Setup phase:** *SO* generates the master key (private data) and the encryption key (public data). The encryption key is distributed to the sensors. Every sensor s gets an access policy Ap_s depending on the kind of data it generates. Finally, *SO* registers the sensors in a transparent list in the blockchain—the *IoT list*. After the setup phase, the operation of the system starts.
1. **Get the revocation policy, generate, encrypt, and sign data:** The sensor s gets a public policy called Revocation Policy, (Rp_i) from the blockchain, generates data, and encrypts them using Rp_i, its policy Ap_s, and a policy that changes with the time. The Slot Attribute (SA_t) gets the ciphertext. Then, it signs the chipertext with its HSM.
2. **Publishing encrypted data and signed digest:** Next, the sensor sends the ciphertext to IPFS and sends the chipertext's hash to the smart-contract *Hash Message Broker*. The smart-contract identifies the sensor before accepting the data through the *IoT list*. The IPFS nodes shall do the same identification to avoid a denial-of-service attack.
3. **Data retrieval:** Knowing the address of the sensor that generates their required data (public information), a *Client*, e.g., Alice, **gets the ciphertext's hash** from *Hash Message Broker* (3.1). Then, using the hash of the ciphertext, Alice **gets the encrypted data** from IPFS (3.2) and checks the integrity of the received ciphertext using the same hash. Next, Alice decrypts the data using her DK. For a successful decryption, Alices's attributes have to satisfy Ap_s, and the Rp_i used when encrypting the message. If the message is old, Alice's attribute must satisfy Ap_s and contain the correct *slot attribute* SA_t.
4. **User revocation:** When a user is revoked, e.g., Bob, *SO* **updates the Revocation Policy** with Rp_{i+1}, removing the decryption capabilities to Bob and all the users in his group. All the non-revoked users in the group, the *affected users*, will require a new DK, updating some of their attributes, the *revocation attributes*, while the remaining ones are unchanged, the *access attributes*.
5. **Decryption key update:** When Alice, the *affected user*, needs a DK update, she **authenticates** to IDP (5.1), which will **provide Alice's identity and attributes** to the *SO* (5.2). The latter needs to provide Alice with a new DK with her *access attributes* and the new *revocation attributes*. Generating a new DK to all *affected users* can take too long if they are a high number. Therefore, to avoid service interruption for the users, the new DKs are already generated, encrypted by the *SO* and stored by the users receiving the name of *backup decryption key*, bDK. The bDK of each user is encrypted with symmetric encryption using a *symmetric key* that is different for each user. Therefore, *SO* will only have to regenerate it and **provide the symmetric key** to Alice (5.3), which is

much faster than generating a decryption key. Next, she can decrypt her bDK, which now will replace her DK. Then, while Alice already has her decryption capabilities recovered, SO generates a new bDK for Alice, which will take a longer time (around 3 min) but will not produce service interruption.

6. Details of the Proposed Scheme

This section will first describe the tools developed for the solution. Then, it will explain the phases in detail, referring to those tools.

6.1. Attributes

An innovative contribution proposed by this research is to take advantage of the infinite attribute universe U offered by the chosen ABE scheme. The attributes can constantly change without changing the master key, encryption, or decryption keys. We define the sets Q, W, and E from the universe U. We assume that Q, W, and E are disjoint subsets of U, i.e., no element in U belongs to more than one of these sets. Then, a client n has a set of attributes $A_n \subseteq Q \cup W \cup E$ with: *access attributes* set $AA = \{A_n \cap Q\}$, *slot attributes* $SA = \{A_n \cap W\}$ and *revocation attributes* $RA = \{A_n \cap E\}$:

- The *access attributes*, AA_n, are those related to the client n position, department, or responsibilities n (e.g., steel_quality_supervisor, efficiency, chemistry_reliability) and can be modified only with SO authorization.
- The *slots attributes*, SA_n, are used to define the periods of the past when the client n had read access. Given a time t_l in Coordinated Universal Time (UTC), its associated *slot*, $slot_l$, can be easily derived $slot_l = t/SlotPeriod$ where $SlotPeriod$ is the duration of every *slot*. Then, from every $slot_l$, a unique attribute $sa_l \in W$ is associated, so $f_{asso} : UTC \rightarrow W$. When a sensor encrypts data at time t_l, it includes the attribute $sa_l = f_{asso}(t_l)$ in the encryption policy so that decryption keys with the attribute sa_l can decrypt the message. SA_n never contains the attribute of current *slot* nor the future *slots*. Therefore, it can only be used to access old encrypted data.
- Each user n is allocated to a group g_i and assigned to either subgroup $subg_{i,A}$ or $subg_{i,B}$. All the users in a subgroup share the same set of *revocation attributes* RA in their corresponding DK, namely $RA_{i,A}$ and $RA_{i,B}$ respectively. However, in their bDK they share a completely different set, $bRA_{i,A}$ and $bRA_{i,B}$. Notice that $RA_{i,A} \cap RA_{i,B} = \{A(g_i)\}$ where $A(g_i)$ is called the *Official revocation attribute* of g_i, $RA_{i,A} \cap bRA_{i,B} = \{ra_j\}$, $RA_{i,B} \cap bRA_{i,A} = \{ra_k\}$ and $bRA_{i,A} \cap bRA_{i,B} = \{ra_l\}$ where $ra_j \neq ra_k \neq ra_l \neq A(g_i)$. This distribution of the *revocation attributes* is used to revoke all the users of a subgroup without affecting their sibling subgroup. It is explained in more detail in the next subsection.

6.2. Revocation

The objective of the revocation is to remove the client's access to any data in the system. However, a malicious client could have downloaded, decrypted, and stored all the messages when he/she was still enrolled in the system. Therefore, perfect revocation is not possible in most cases, so the main purpose of the proposed scheme is to prevent the revoked client from accessing new data.

The user revocation in our system is performed by mixing indirect revocation and direct revocation [50]. In direct revocation, there is a list of revoked users, and the writers need to be aware of this list while encrypting to exclude the revoked users. In indirect revocation, all the users but the revoked user (all users are *affected users*) need to contact the key authority to get the update of their decryption keys. In our solution, we use both indirect and direct revocation.

To apply indirect revocation, we propose modifying one attribute, the *official revocation attribute*, which is present and required in all the policies of the ABE encryption. Then all the users need a new decryption key with the new attribute. With this system, there is no need to perform a new setup and change the PK for all the writers which can be

annoying if there are too many IoT devices. However, indirect revocation still can be too computationally expensive for *SO* if the number of users that need a key update (*affected users*) is high. To avoid revoking all the users with every revocation, which would require many key updates, we apply direct revocation by dividing the users into groups, e.g., eight groups, where each group has an *official revocation attribute*. Then, during the encryption, the *official revocation attributes* of all the groups are included and combined with *or* gates so all the groups can decrypt the message. It still takes long to provide new keys to all the users in a group when there is a revocation. This problem would lead to a service interruption for all the *affected users*. To solve this problem, we exploited the concept of *backup decryption key*, *bDK*. All the users already have the new decryption key with their group's next *official revocation attribute*, the *bDK*, but it is encrypted using symmetric encryption. So, only providing the affected users with the symmetric key is enough to recover their decryption capabilities. All the *bDK* are encrypted using a different symmetric keys, so the revoked users cannot use the symmetric key of any other *affected user*, avoiding a collusion revocation attack. Since every symmetric key is only useful for the recipient users, the list with the symmetric keys for the *affected users* can be shared with gossip protocol [51] to avoid too many petitions to the *SO* and its size will be around 16 MB for 1,000,000 *affected users*.

Still, with this mechanism, the revocation policy has the size of the number of groups. Thus, this system uses what here is called subgroups, to reduce the revocation policy size to half of the number of subgroups. Notice that under each group, there are two sibling subgroups that are linked—subgroup A and subgroup B ($sg_{i,A}$ and $sg_{i,B}$). Given the distributions of the *RA* explained in the last subsection, there is always a shared *ra* in the *DKs* of the subgroups $sg_{i,A}$ and $sg_{i,B}$, the $A(g_i)$, which is the one included in *Rp* to cover both subgroups. For example, when the user Bob from $sg_{i,B}$ is revoked, all users in $sg_{i,B}$ but Bob will get access to their *bDK* with $bRA_{i,b}$, then the new $A(g_i)$ will pass to be ra_j where $ra_j = RA_{i,A} \cap bRA_{i,B}$, and *Rp* will be modified to replace the old $A(g_i)$ with the new $A(g_i)$. So Alice from $sg_{i,A}$ with $RA_{i,A}$ will be able to satisfy the new *Rp* without the need to update her *DK*. All users in $sg_{i,b}$ will get a new *bDK* with a new $bRA_{i,b}$. The new $bRA_{i,b}$ will share a new *ra* with $RA_{i,A}$ and another with $bRA_{i,A}$, which requires *ra*, which will not have been shared previously from $RA_{i,A}$ and another in $bRA_{i,A}$. This mechanism has a limitation, as every time a user from a subgroup, e.g., $sg_{i,B}$, is revoked, the $RA_{i,B}$ is revoked, so its attributes become useless, and the *ra* shared with $bRA_{i,A}$ becomes useless too. Therefore, being $bRA_{i,A} = \{ra_1, ra_2...ra_x\}$, it will become completely useless after x revocations of $RA_{i,B}$ if $bRA_{i,A}$ is not revoked earlier, i.e., there are not two revocations in $sg_{i,A}$. Equation (1) defines the probability, denoted as P_r, of experiencing x revocations in one subgroup before encountering two revocations in the sibling subgroup. We recommend distributing 10 *ra* in each *RA*, highly reducing the probability of this event to happen as defined in Equation (1). Still, this event would simply require updating also the *bDK* of the clients in $sg_{i,A}$ when there is a revocation in $sg_{i,B}$.

$$P_r = \frac{2(x+1)+2}{2^{(x+1)}} \tag{1}$$

With this solution, we mix both indirect and direct revocation to reduce the revocation list to a list of eight *revocation attributes* and to reduce the *affected users* to a 16th of the total users.

Notice that this mechanism is only needed for non-planned revocation. Nevertheless, usually, the revocations are planned, e.g., the contract of a worker or company finishes. Planned revocation does not require a short time slot, i.e., they can be revoked within the day. Therefore, all the planned revocations of a subgroup in the day are performed at once, highly reducing the impact on the *SO*. Moreover, they are preferably performed when there is a non-planned revocation in the subgroup within the day, which would avoid including any overhead in the *SO*.

Proof of Revocation

Revocation means that an adversary Adv_A revoked at time t will not be able to decrypt any new message published after time t. This property requires that even if the adversary has the AA that satisfy the Access policy of a message, such a message cannot be decrypted. To prove that our solution for revocation securely revokes Adv_A, we also consider an adversary Adv_B. Adv_A and Adv_B are trying to distinguish two messages given the cipher text using a set of decryption keys that does not satisfy the policies. The objective of this proof is to show that Adv_A has the same advantage to break the solution as Adv_B to break the ABE construction of Waters [43].

- Theorem: Suppose that the assumptions of the ABE construction holds. Then no adversary with a revoked key at time t can distinguish a ciphertext from one of two messages encrypted after time t with non-negligible advantage.
- Setup: Challenger runs the Setup() function, and share ABE-PK with Adv_A and Adv_B. Challenger defines a published set of attributes $ARp \subset E$ and a policy RP such that RP is the *or* combination of all the elements of ARP. In addition, Challenger defines two sets of attributes $RA_{1,A} \subset E$ and $bRA_{1,A} \subset E$ such that $RA_{1,A} \cap ARp = \emptyset$ and $|bRA_{1,A} \cap ARp| = 1$ as defined in Section 6.1. Notice that with this given configuration $Y(RP, RA_{1,A}) = 0$ but $Y(RP, bRA_{1,A}) = 1$.
- Query phase: Adv_A request to Challenger j different DKs, $\{DK_j\}_{j\in\mathbb{N}}$ associated j set of Access Attributes $\{AA_j \subseteq Q\}$ and j sets of Slot Attributes $\{SA_j = \{w \in W | w < sa_c\}\}$. Notice that Adv_A cannot include in its query attributes from the subset E and therefore any of these DKs can satisfy RP. Adv_B request to Challenger i different DKs, $\{DK_i\}_{i\in\mathbb{N}}$ associated with i sets of attributes $\{S_i\}$.
- Response phase: Challenger provides to Adv_A the j different DKs with the associated attributes $\{AA_j \cup SA_j \cup RA_{1,A}\}$. Challenger provides to Adv_B the i differents DKs.
- Challenge: Adv_A prepares two equal length messages M_{A1} and M_{A2}. Then it gives the two messages and Access Policy Ap to Challenger. Challenger flips a coin c and encrypts M_{Ac} with policy $P = (Ap) \wedge (sa_c \vee Rp)$. The resulting ciphertext CT_A is given to Adv_A. Notice that $Y(P, AA_j \cup SA_j \cup RA_{1,A}) = 0 \ \forall j$. Adv_B prepares two equal length messages M_{B1} and M_{B2}. Then it gives the two messages and a policy $P : Y(P, S_k) = 0 \forall k \leq i$. Challenger flips a coin c and encrypts the M_{Bc} with the policy P and returns the cipher text to Adv_B.
- Guess: The Adv_A uses a function f_A that, given $\{Dk_j\}\forall j$ and the encrypted message CT_A, makes a guess of M' with a non-negligible advantage (ϵ_A), as shown in Equation (2):

$$Pr[f_A(\{Dk_j\}\forall j, CT_A) = Ma_c] = \frac{1}{2} + \epsilon_A \tag{2}$$

The Adv_B then uses a function f_B that given $Dk_i \forall i$ and the encrypted message CT_B makes a guess of M' a non-negligible advantage (ϵ_B), as shown in Equation (3):

$$Pr[f_B\{Dk_i\}\forall i, CT_B) = Mb_c] = \frac{1}{2} + \epsilon_B \tag{3}$$

However, Adv_A cannot satisfy the policy P because $(AA_j \cap E = \emptyset) \wedge (SA_j \cap E = \emptyset) \wedge (sa_c \notin SA_j) \wedge (RA_{1,A} \cap ARp = \emptyset) => Y((sa_c \vee Rp), (AA_j \cap SA_j \cap RA_{1,A})) = 0$. Therefore, Adv_A and Adv_B are in the same scenario. If f_a exists, it could be used as a substitute of f_b by Adv_B to get a non-negligible advantage. However, the ABE construction used proves that f_b does not exist, and therefore neither f_a. In this way, we prove that Adv_A cannot get access to new messages after RP is updated at time t without first accessing $bRA_{1,A}$.

6.3. Smart Contracts

This section defines the smart contract interfaces and the algorithm logic. There are three smart contracts: Revocation Policy Smart Contract, IoT List, and Hash Message Broker.

- Revocation Policy Smart Contract (RPSM): a smart contract that holds the Revocation policy. All the participants in the blockchain can read this policy; however, only the *SO* can modify the policy. Section 6.2 provides more details about the revocation policy and its management.
- IoT list: a smart contract that stores the addresses of all the sensors authorized by the *SO* to send information to the system. Only the *SO* can modify this list, but all the members in the blockchain can read it. A transparent list in the blockchain is essential for the chain of trust to provide strong authentication guarantees of the sensors to the clients. More details are provided in Section 7.2.
- Hash Message Broker (HMB): this smart contract links every sensor's address with the hash of its messages and its metadata. When receiving a transaction from a sensor to store a hash, the smart contract queries the *IoT list* for the address of the sender. If the sender of the transaction is in the list, the hash and its metadata are stored linked to the address of the sender.

6.4. Phases

The proposed scheme is logically divided into four phases: set up, sending data, reading data, and updating the decryption keys.

6.4.1. Set Up (*SO* → *Sensor, IoT List, RPSM*)

This phase includes all the processes that should be executed one time per system or per sensor. Firstly, from a random seed, *SO* generates the MSK and ABE-PK. The random seed is deleted, and the MSK is kept secret. In addition, the *slot* duration, *SlotPeriod*, is defined (usually 24 h). When a new sensor s is to be included in the system, *SO* registers the s's address in the smart contract *IoT list*, provides it with the ABE-PK, and defines the access policy Ap_s for the encrypted data of this *Sensor*. Finally, the *SO* uploads a Rp_i to RPSM. The Rp_i is a policy formed by the *Official revocation attributes* of all the groups united by *or*.

6.4.2. Sending Data (*Sensor* → *IPFS, HMB*)

A sensor s generates a message M_i at a particular time t_i. Then from t_i the sensor gets the $sa_i \in W$, from the Revocation policy Smart Contract it gets the Revocation policy Rp_i, and performs the encryption as shown in Equation (4):

$$EM_i = Encrypt(ABE-PK, (Ap_s) \wedge (sa_i \vee Rp_i), M_i) \quad (4)$$

Then the encrypted message, EM_i, is signed and sent to IPFS. The last one can check the sender's validity in the *IoT list*. Meanwhile, the s uses the same credentials to send a transaction to the smart contract *Hash message broker* with the hash of the EM_i, $Hash_i$. If and only if the address of the IoT device is in the *IoT list*, $Hash_i$ and the M_i metadata are accepted, and then they are linked to the s' address, including the encrypted message in the set of messages from s.

6.4.3. Reading Data (*HMB,IPFS* → *Alice*)

The client Alice (a) needs data from a *Sensor* s. Knowing the address of the device, she can query the metadata of their messages and use it to identify the required message M_r. Alice tries to decrypt it using her decryption key DK, ($Decrypt(EM_r, DK)$). If M_r is a message from the current slot sa_c, the decryption will be successful if and only if, $Y(Rp_r, RA_a) \wedge Y(Ap_s, AA_a)$. However, when decrypting a non-current message from the slot sa_r, the decryption will be successful if and only if $sa_r \in SA_a \wedge Y(Ap_s, AA_a)$.

6.4.4. User Revocation: (*SO* → *RPSM*)

When a user Bob from the group i and subgroup $subg_{i,B}$ is revoked, *SO* has to update the Rp in the RPSM. Particularly *SO* has to update the $A(g_i)$. The new $A(g_i)$ will be the

only *ra* that is in the *RA* of the *bDK* of all the users of $subg_{i,B}$ and that it is at the same time in all the *RA* of the *DK* of all the users of $subg_{i,A}$, as shown in Equation (5):

$$A(g_i) = bRA_{i,B} \cap RA_{i,A} \tag{5}$$

This way, Bob and all the users in $subg_{i,B}$ will not be able to decrypt any new data. All the affected users in $subg_{i,B}$ will receive access to their *bDK*—apart from Bob.

6.4.5. Updating Decryption Key (*SO* → *Alice*)

When a user is revoked, the *Rp* is updated, changing the *Official revocation attribute* of the group of the revoked user, which may make Alice lose her capability to decrypt new data. To avoid this, Alice will quickly get a symmetric key from *SO* or another client through a gossip protocol [51]. With this symmetric key, she will be able to decrypt her *bDK*, which has the new $A(g_i)$. In this moment, Alice starts to authenticate herself to *IDprovider* in order to connect with *SO* and request a new *bDK*, which will be encrypted with a new symmetric key.

7. Evaluation

The prototype was deployed in a relevant environment for evaluation purposes. This test environment is a flexible cyber-physical system composed of specific hardware (i.e., industrial gateways and Raspberry PIs) and is extensible with virtual devices hosted on a server with 28 cores and 256 Gb of RAM. The laboratory is configured using VLANs to represent all the different layers in an Industry 4.0 architecture, as presented in Figure 1.

The prototype services are distributed on different devices (whether physical or virtual): The sensors were implemented using Raspberry Pi 4B [52] equipped with a real HSM, Iridium 9670–TPM2.0 chip [53] to perform secure crypto-operations. On the other hand, services which in a real scenario are hosted on IT network are deployed on virtual machines: the IPFS servers were distributed on four separate virtual machines, and blockchain nodes were implemented with Hyperledger Fabric 1.4 [46] in three other virtual machines, all of them with Ubuntu 20.04 [54] and 4 Gb of RAM.

7.1. Performance Evaluation

The proposed scheme requires a continuous update of the decryption keys, which can lead to a bottleneck in the decryption key provider, the *SO*. In this section, an analysis of the scalability of the proposed system is presented, considering the number of users as a scale-up factor. As explained in Section 6.2, the revocations are commonly planned and they do not affect the performance. However, in non-planned revocations, a group of non-revoked users lose access to data, requiring a key update, which can lead to a bottleneck in the *SO*. In the following we present the analysis results conducted by utilizing standard hardware to verify that no bottleneck is measured in practice.

Let us define *Nu* as the total number of users, *Nsg* as the number of subgroups and, *NuG* as the number of users per subgroup. In addition, consider *Pv* as the average time a user needs a non-planned revocation, and V_G as the number of *DK* that *SO* can generate every second.

To avoid *SO* overhead, the time of generating the *bDK* of a subgroup shall be less than the average time there is a revocation in the system. Equations (6)–(8) show the estimation of the maximum number of users in the proposed solution.

$$\frac{NuG}{V_G} \leq \frac{Pv}{Nu} \tag{6}$$

$$\frac{Nu}{Nsg * V_G} \leq \frac{Pv}{Nu} \tag{7}$$

$$Nu \leq \sqrt{Pv * Nsg * V_G} \tag{8}$$

Using the library OpenABE [55] in a laptop Thinkpad E495 with 16 GB RAM, AMD Ryzen 7 3700 and Radeon Vega Mobile GFX 8 [56], $Pv \geq 600$ DK/s, with 30 attributes in each DK, are observed. Using a Rp of 8 *official revocation attribute* united with *or* gates, it results in $Nsg = 16$. Pv can vary from scenario to scenario; nevertheless, assuming a similar behavior of X.509 certificates estimated by NIST [57], a 10% of key revocations are considered. This means that if a key has a lifetime of 3 months, every user needs a key revocation every 30 months on average. As a result, this system could manage 864,000 users with the computational power of a laptop. In addition, based on the estimation in Equation (8), by using a server 7 times as powerful as the test laptop the system could reach 2,000,000 users.

The analysis performed highlights that the solution has a limited scalability, as the number of maximum clients is proportional to the square root of the computational power. Therefore it could not be deployed to applications with massive user numbers such as Instagram or Telegram (which have more than 500 million users each). However, it can be considered applicable for most professional manufacturing environments (i.e., Infineon Technologies) given that only one laptop could manage the credentials of the workers of an entire company (50,000+ workers).

7.2. Security Evaluation

This section evaluates a set of security properties that the proposed solution is guaranteeing. In the first place, it presents the common security features guaranteed in most ABE-based infrastructures:

- Collision resistance: the most basic requirement for fine-grained authorization. All the sensitive information is encrypted with ABE mechanism, which provides collision resistance.
- End-to-end Encryption: the data is encrypted by the sensors using ABE and it is only decrypted by the clients with the correct DK's client. Additionally, unlike in Refs. [17,38], the proposed system does not trust the blockchain or IPFS with any sensitive information that could lead to a data leak.
- Forward secrecy: "The newly joined user can also decrypt the previously published ciphertexts if it has sufficient attributes" [31]. All the messages are encrypted including in the policy a *slot attribute*. Then, in the future, when a user (Alice) wants to access some old data from a particular $slot_l$, she simply asks the SO for a DK with her access attributes and sa_l.

In the following, we outline the essential security properties that are not guaranteed by all of the analyzed works in the state-of-the-art, while these properties are integral to our proposed solution:

- User Revocation (UR): the revocation of a user is performed by modifying a policy published in a smart contract on the blockchain. Once it gets published, it is applied to all the encryption processes, revoking a portion of the users. To avoid this portion of the users in the system losing their decryption capabilities when they should not be revoked *backup decryption keys*, the affected users get access to a symmetric key to decrypt their *backup decryption keys*, thus avoiding service interruption.
- Collision Resistance Revocation (CRR): as explained in Section 3, a revoked user could collaborate with a non-revoked user to get access to data that shall be inaccessible for the collaborators individually. In the proposed system, every *backup decryption key* is encrypted using a unique symmetric key. Therefore, the information needed to recover an affected user's data access (the symmetric key) is useless for any other user. Thus, a revoked user cannot collaborate with other users.
- Data Integrity (DI): defined here as the capacity of the client to verify the non-modification of the data. In the proposed system, the client does not need to verify any signature (apart from those needing to connect with the blockchain) in order to verify the data integrity. All data hashes are stored in the immutable blockchain;

therefore, even if the data is stored in a non-trusted service such as IPFS, this data is as immutable as the blocks in the blockchain.

- No Sensitive Data in the Blockchain (NSDB): as explained in Section 3, sensitive data shall never be exposed to the blockchain. In our solution, no private keys or partial decryption keys are stored in the blockchain. The only public information is the identities of the IoT devices, in the *IoT list*, and the metadata needed to identify the required messages.

While the data authentication feature may be a straightforward task in ABE solutions involving human or organizational writers, the authentication of data in ABE applied to IoT devices requires further evaluation:

- Data authenticity: the capacity to securely identify the data generator entity. This means identifying that specific data in the system comes exactly from a particular IoT node. To do so, the smart contract *Hash message broker* only accepts data hashes sent in a signed transaction. From the signed transaction, the smart contract gets the sender's address and stores the hash together with the address if the address is approved in the smart contract *IoT list*. An address that is approved on the *IoT list* means that the public key utilized to generate the address has been verified to be associated with the Hardware Security Module of a specific IoT device. Although this paper does not cover this verification process, it can be accomplished using Platform Certificates part of the standard *Trusted Platform Module* [48] or using IDevID certificates of the standard *IEEE Std 802.1AR: Secure Device Identity* [58]. With this chain of trust, the data origin is as reliable and secure as the *Hardware Security Module* of the IoT device.

7.3. Requirements Evaluation

Table 1 explains how the proposed solution covers the necessities of secure data sharing identified in the industrial use case based on the technology performance and security evaluation.

Table 1. Evaluation of the requirements.

Requirement	Evaluation
RQ1	The system provides *end-to-end encryption*, which guarantees data confidentiality protection both in transit and at rest.
RQ2	Using ABE, the data access of the client can be defined beforehand through a combination of attributes and the data can be encrypted by defining a policy that can decrypt it. This way, it is possible to define fine-grained data access for each user. Additionally, given the *collision resistance*, different users cannot collaborate to access more data.
RQ3	There are two mechanisms used to limit the time access to users, non-planned revocation and planned revocation, which are both explained in Section 6.2. This way, the reading privileges of a user can be removed or modified whenever required.
RQ4	Because of the *data integrity* of the system based on blockchain, the information provided in the system by the IoT nodes has integrity guarantees to be used as proof of certification.
RQ5	As explained in Section 7.2, this system has limited scalability with the number of users; however, with basic resources, it can be applied to 800,000 users and could easily be increased to 2,000,000, which covers the majority of industrial scenarios.
RQ6	The proposed solution has a chain of trust ending in the reliability of the HSM in order to provide authenticity guarantees to the client. This is explained in detail in Section 7.2.
RQ7	The hash of all data and its metadata are stored in the blockchain in an immutable fashion, so even if critical data with high impact is intentionally deleted from the IPFS, any party that accessed the data can claim its veracity.

8. Discussion and Future Work

In this paper we have proposed a robust mechanism for many-to-many data sharing with security-by-design. In this solution, we have considered and covered the six well-known security threats of the STRIDE methodology: Spoofing, Tampering, Repudiation, Information disclosure, Denial of service, and Elevation of privilege [59]. To achieve this result, many different technologies had to be combined (Blockchain, ABE, HSM, IPFS, OpenID), and some even created (Revocation). In the end, we ended up with a complex system that delivers the necessary security properties for data sharing with fine-grained authorization: Authenticity, Integrity, Non-repudiation, Confidentiality, Availability, and Authorization.

Despite our achievement, there are other results in the state-of-the-art that offer solutions to many of the mentioned security threats, but as shown in Table 2, none of them fulfill them completely. It is important to consider that information security solutions do not follow a linear progression, in which each solution is incrementally stronger than the previous one. Instead, they resemble a chain, where the overall security is only as strong as its weakest link, as stated by Thomas Finne [60]. For this reason, we consider the current work to be a great leap forward from previous results as it finally offers a complete security solution for data sharing with fine-grained authorization supporting revocation, which is the long-standing problem of ABE. Such a result enables many new functionalities requiring data sharing that were previously impossible to achieve due to security reasons or because they were too tedious. For example, it greatly enhances the implementation of machine learning in smart factories, because one of the most challenging tasks in this technology is data collection [61], and this task can be even more complicated in the industry field due to the sensitivity of the information.

Table 2. Comparison of blockchain-based ABE protocols.

ABE and	**Security Properties**			
Blockchain	UR	**CRR**	**DI**	**NSDB**
[18]	X	X	✓	✓
[34]	X	X	✓	✓
[35]	X	X	✓	✓
[36]	X	X	X	✓
[37]	X	X	NA	✓
[38]	X	X	X	X
[17]	NA	NA	X	X
[42]	✓	X	✓	✓
Proposed solution	✓	✓	✓	✓

Even if our own work fits into most of the Industrial scenarios, it is far from perfectly applicable to overall scenarios. As explained in Section 7.1, it is necessary to prepare a server acting as a key provider (*SO*) with a capability proportional to the square of the clients. So, while it is easily applicable to more than a million clients, it would be almost impossible to apply this solution to a billion clients, such as those scenarios where IoT nodes also read data from the share data-based or in a global social network.

In addition, there are many additional ABE features, which although not necessary for a fully secure architecture and for our scenario, may provide more flexibility and applicability to the solution. From a Survey of attribute-based encryption, [62], we can discover the following ABE features that our solution does not have: Decentralized Authority, Asynchronous Authority, Key Delegation, and Privacy Preservation.

However, these missing features are not intrinsically unfeasible in our architecture—they were simply out of scope. For example, OpenID offers the possibility to identify the attributes of a user without revealing the user's identity for Privacy Preservation. It also

provides the possibility to identify a client through several ID providers decentralizing attribute issuers. On the other hand, our solution is designed on the Waters construct, but there are many other ABE constructs with different features that could be applied as long as they allow infinite attributes. Additionally, Key Delegation could be performed in our solution by making the blockchain witness the operation to solve the system's accountability problem defined in Ref. [62]. Finally, the trustworthiness of the IoT devices is not covered in this work, nor is data re-encryption, hence, technologies such as remote attestation and Chameleon Hash-based blockchains are valid future works for this article. Therefore, there are many options for further research and adaptation of the solution to other scenarios with different requirements.

9. Conclusions

In this article, we present a real scenario where data generated from multiple IoT devices can be consumed by different stakeholders in a collaborative ecosystem motivated by the transition to Industry 4.0. We highlight the necessity of data authentication, integrity, non-repudiation, and confidentiality with fine-grained access control. Firstly, IPFS is used to provide data availability and scalability. We use blockchain to protect the metadata and data hashes, which is essential for the integrity protection of the data in IPFS. Then, the HSMs offer strong guarantees of the authenticity of the data generators and we create a smart contract IoT list to assure the correct identity of each HSM transparently and we implement OpenID to improve the dynamism of client enrollment in the system. Finally, we propose a novel solution to revoke users of ABE, using revocation attributes to reduce the impact of indirect revocation and backup decryption keys to delete any possible side effect of the revocation on the legitimate users. In a performance evaluation, we prove that the solution can be easily applied to 2,000,000 users, which satisfies any industrial scenario. With our work, we demonstrate that controlled data sharing with all the security guarantees from many IoT devices to many clients is possible. This solution has the potential to make a significant impact in collaborative manufacturing and thus in the Industry 4.0. Our hope is that this investigation enables further research and solutions in the field of secure data sharing and ultimately forms part of the realization of revolution in the manufacturing process.

Author Contributions: Conceptualization, E.G.-M., D.M. and V.S.; methodology, E.G.-M.; software, E.G.-M.; validation, E.G.-M., D.M. and V.S.; formal analysis, E.G.-M.; investigation, E.G.-M.; writing—original draft preparation, E.G.-M., D.M. and V.S.; writing—review and editing, E.G.-M., D.M., V.S., E.C. and L.P.; visualization, E.G.-M., D.M., V.S., E.C. and L.P.; project administration, E.G.-M. All authors have read and agreed to the published version of the manuscript.

Funding: This research has been founded by the European Union's Horizon 2020 Research and Innovation program under grant agreement No. 871518, a project named COLLABS [19].

Acknowledgments: We are grateful to Cyrille Martins for the help implementing OpenID in the system. However, any errors or problems found in the present work are solely and exclusively the responsibility of the authors of this document.

Conflicts of Interest: The authors declare no conflict of interest.

Abbreviations

The following abbreviations are used in this manuscript:

IoT	Internet of Things
IIoT	Industrial Internet of Things
CM	Collaborative Manufacturing
IPFS	InterPlanetary File System
ABE	Attribute-Based Encryption
PL	Purdue Level
CP-ABE	Ciphertext-Policy Attribute-Based Encryption

URL	Uniform Resource Locator
HSM	Hardware Security Modules
CT	CipherText
TPM2.0	Trusted Platform Module 2.0
UTC	Coordinated Universal Time
UR	User Revocation
Q	Set of attributes that contains all the *access attributes*
W	Set of attributes that contains all the *slots attributes*
E	Set of attributes that contains all the *revocation attributes*
AA_n	Set of attributes that contains all the *access attributes* of client n
SA_n	Set of attributes that contains all the *slots attributes* of client n
DK	Decryption key of ABE, it has attributes assigned and it is unique per client
bDK	Backup of a DK, it is encrypted with an unique symmetric key
g_i	Group i
$A(g_i)$	*Official revocation attribute* of g_i
$subg_{i,A}$	First subgroup of group i
$subg_{i,B}$	Second subgroup of group i
$RA_{i,A}$	Set of *revocation attributes* of clients' DK in $subg_{i,A}$
$bRA_{i,A}$	Set of *revocation attributes* of clients' bDK in $subg_{i,A}$
t_n	Particular time n
sa_i	Slot attribute used in encryption at t_i
$ABE\text{-}PK$	Public key needed to perform ABE encryption
MSK	Master Secret Key of ABE, needed to create DK and PK
SO	Sensor Owner, responsible for the management of ABE and the only entity with knowledge of MSK
IDP	ID provider, identifies the client and their attributes to SO using OpenID
HMB	Hash Message Broker smart contract
R_p	Revocation policy, combination of all the $A(g)$ united with *or* logic functions
$RPSM$	Revocation Policy smart contract
Adv	Adversary
Nu	Number of clients in the system
NuG	Number of clients per group
Pv	Average time a client needs a non-planned revocation
V_G	Number of DK that SO can generate every second
CRR	Collision Resistance Revocation
DI	Data Integrity
NSDB	No Sensitive Data in the Blockchain

References

1. Elijah, O.; Ling, P.A.; Abdul Rahim, S.K.; Geok, T.K.; Arsad, A.; Kadir, E.A.; Abdurrahman, M.; Junin, R.; Agi, A.; Abdulfatah, M.Y. A Survey on Industry 4.0 for the Oil and Gas Industry: Upstream Sector. *IEEE Access* **2021**, *9*, 144438–144468. [CrossRef]
2. Lasi, H.; Fettke, P.; Kemper, H.G.; Feld, T.; Hoffmann, M. Industry 4.0. *Bus. Inf. Syst. Eng.* **2014**, *6*, 239–242. [CrossRef]
3. Kim, J.H. A review of cyber-physical system research relevant to the emerging IT trends: Industry 4.0, IoT, big data, and cloud computing. *J. Ind. Integr. Manag.* **2017**, *2*, 1750011. [CrossRef]
4. Soori, M.; Arezoo, B.; Dastres, R. Internet of things for smart factories in industry 4.0, a review. *Internet Things Cyber-Phys.Syst.* **2023**, *13*, 192–204. [CrossRef]
5. Technology Business Research Company. *IoT in Manufacturing Global Market Report 2023*; ResearchAndMarket U.S.: Hampton, NH, USA, 2023.
6. Betti, F.; Bezamat, F.; Fendri, M.; Fernandez, B.; Küpper, D.; Okur, A. Share to Gain: Unlocking Data Value in Manufacturing. World Economic Forum, 2020. Available online: https://www3.weforum.org/docs/WEF_Share_to_Gain_Report.pdf (accessed on 18 May 2023).
7. Yu, Y.; Chen, R.; Li, H.; Li, Y.; Tian, A. Toward Data Security in Edge Intelligent IIoT. *IEEE Netw.* **2019**, *33*, 20–26. [CrossRef]
8. Müller, J.M.; Veile, J.W.; Voigt, K.I. Prerequisites and incentives for digital information sharing in Industry 4.0—An international comparison across data types. *Comput. Ind. Eng.* **2020**, *148*, 106733. [CrossRef]
9. Irdeto. *Global Connected Industries Cybersecurity Survey*; Irdeto: New Delhi, India, 2019.
10. Waraga, O.A.; Bettayeb, M.; Nasir, Q.; Talib, M.A. Design and implementation of automated IoT security testbed. *Comput. Secur.* **2020**, *88*, 101648. [CrossRef]

11. Sarker, I.H.; Khan, A.I.; Abushark, Y.B.; Alsolami, F. Internet of things (IoT) security intelligence: A comprehensive overview, machine learning solutions and research directions. *Mob. Netw. Appl.* **2022**. [CrossRef]
12. Abiodun, O.I.; Abiodun, E.O.; Alawida, M.; Alkhawaldeh, R.S.; Arshad, H. A review on the security of the internet of things: Challenges and solutions. *Wirel. Pers. Commun.* **2021**, *119*, 2603–2637. [CrossRef]
13. Amazon Web Services. *Summary of the Amazon S3 Service Disruption in the Northern Virginia (US-EAST-1) Region*; Amazon Web Services, Inc.: Seattle, WA, USA, 2017. Available online: https://aws.amazon.com/message/41926/ (accessed on 18 May 2023).
14. Moss, S. *Microsoft Azure Suffers Outage after Cooling Issue*; DCD. 2018. Available online: https://www.datacenterdynamics.com/en/news/microsoft-azure-suffers-outage-after-cooling-issue/ (accessed on 18 May 2023).
15. Fu, Y. *Alibaba Cloud Reports IO Hang Error in North China*; Shanghai EqualOcean Technology Co., Ltd.: Shanghai, China, 2019. Available online: https://equalocean.com/news/201903031507 (accessed on 18 May 2023).
16. Judge, P.; Swinhoe, D. *Cooling Failure Brings Down Google Cloud Data Center in London on UK's Hottest Day*; DCD 2022. Available online: https://www.datacenterdynamics.com/en/news/cooling-failure-brings-down-google-cloud-data-center-in-london-on-uks-hottest-day/ (accessed on 18 May 2023).
17. Sharma, P.; Jindal, R.; Borah, M.D. Blockchain-based cloud storage system with CP-ABE-based access control and revocation process. *J. Supercomput.* **2022**, *78*, 7700–7728 [CrossRef]
18. Wang, S.; Zhang, Y.; Zhang, Y. A blockchain-based framework for data sharing with fine-grained access control in decentralized storage systems. *IEEE Access* **2018**, *6*, 38437–38450. [CrossRef]
19. COLLABS-871518 Project Website. Available online: https://www.collabs-project.eu/ (accessed on 18 May 2023).
20. Benet, J. Ipfs-content addressed, versioned, p2p file system. *arXiv* **2014**, arXiv:1407.3561.
21. OFoundation. OpenID. Available online: https://openid.net/ (accessed on 18 May 2023).
22. Williams, T.J. The Purdue enterprise reference architecture. *Comput. Ind.* **1994**, *24*, 141–158. [CrossRef]
23. Lienberherr, K. Formulations and Benefits of the Law of Demeter. *ACM SIGPLAN Not.* **1989**, *24*, 67–78. [CrossRef]
24. COLLABS Consortium. *D1.2 COLLABS System Architecture Definition*; COLLABS EU-Project, Europe 2022. Available online: https://www.collabs-project.eu/wp-content/uploads/2022/07/D1.2.pdf (accessed on 18 May 2023).
25. Hwang, Y.H.; Lee, P.J. Public key encryption with conjunctive keyword search and its extension to a multi-user system. In Proceedings of the Pairing-Based Cryptography–Pairing 2007: First International Conference, Tokyo, Japan, 2–4 July 2007; Springer: Berlin/Heidelberg, Germany, 2007; pp. 2–22.
26. Sahai, A.; Waters, B. Fuzzy identity-based encryption. In Proceedings of the Advances in Cryptology—EUROCRYPT 2005: 24th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Aarhus, Denmark, 22–26 May 2005; Springer: Berlin/Heidelberg, Germany, 2005; pp. 457–473.
27. Bethencourt, J.; Sahai, A.; Waters, B. Ciphertext-Policy Attribute-Based Encryption. In Proceedings of the 2007 IEEE Symposium on Security and Privacy (SP '07), Oakland, CA, USA, 20–23 May 2007; pp. 321–334. [CrossRef]
28. Albulayhi, K.; Abuhussein, A.; Alsubaei, F.; Sheldon, F.T. Fine-grained access control in the era of cloud computing: An analytical review. In Proceedings of the 2020 10th Annual Computing and Communication Workshop and Conference (CCWC), Las Vegas, NV, USA, 6–8 January 2020; pp. 748–755.
29. Li, M.; Yu, S.; Zheng, Y.; Ren, K.; Lou, W. Scalable and secure sharing of personal health records in cloud computing using attribute-based encryption. *IEEE Trans. Parallel Distrib. Syst.* **2012**, *24*, 131–143. [CrossRef]
30. Yu, S.; Ren, K.; Lou, W. FDAC: Toward Fine-Grained Distributed Data Access Control in Wireless Sensor Networks. *IEEE Trans. Parallel Distrib. Syst.* **2011**, *22*, 673–686. [CrossRef]
31. Yang, K.; Jia, X. Expressive, Efficient, and Revocable Data Access Control for Multi-Authority Cloud Storage. *IEEE Trans. Parallel Distrib. Syst.* **2014**, *25*, 1735–1744. [CrossRef]
32. Tysowski, P.K.; Hasan, M.A. Hybrid Attribute- and Re-Encryption-Based Key Management for Secure and Scalable Mobile Applications in Clouds. *IEEE Trans. Cloud Comput.* **2013**, *1*, 172–186. [CrossRef]
33. Hur, J.; Noh, D.K. Attribute-based access control with efficient revocation in data outsourcing systems. *IEEE Trans. Parallel Distrib. Syst.* **2010**, *22*, 1214–1221. [CrossRef]
34. Qin, X.; Huang, Y.; Yang, Z.; Li, X. LBAC: A lightweight blockchain-based access control scheme for the internet of things. *Inf. Sci.* **2021**, *554*, 222–235. [CrossRef]
35. Guan, Z.; Lu, X.; Yang, W.; Wu, L.; Wang, N.; Zhang, Z. Achieving efficient and Privacy-preserving energy trading based on blockchain and ABE in smart grid. *J. Parallel Distrib. Comput.* **2021**, *147*, 34–45. [CrossRef]
36. Zhang, Y.; He, D.; Choo, K.K.R. BaDS: Blockchain-based architecture for data sharing with ABS and CP-ABE in IoT. *Wirel. Commun. Mob. Comput.* **2018**, *2018*, 2783658. [CrossRef]
37. Pournaghi, S.M.; Bayat, M.; Farjami, Y. MedSBA: A novel and secure scheme to share medical data based on blockchain technology and attribute-based encryption. *J. Ambient Intell. Humaniz. Comput.* **2020**, *11*, 4613–4641. [CrossRef]
38. Jemel, M.; Serhrouchni, A. Decentralized access control mechanism with temporal dimension based on blockchain. In Proceedings of the 2017 IEEE 14th International Conference on e-business Engineering (ICEBE), Shanghai, China, 4–6 November 2017; pp. 177–182.
39. Zhang, R.; Xue, R.; Liu, L. Security and privacy on blockchain. *ACM Comput. Surv. (CSUR)* **2019**, *52*, 1–34. [CrossRef]

40. Horváth, M. Attribute-based encryption optimized for cloud computing. In Proceedings of the SOFSEM 2015: Theory and Practice of Computer Science: 41st International Conference on Current Trends in Theory and Practice of Computer Science, Pec pod Sněžkou, Czech Republic, 24–29 January 2015; Springer: Berlin/Heidelberg, Germany, 2015; pp. 566–577.
41. Liang, X.; Li, X.; Lu, R.; Lin, X.; Shen, X. An efficient and secure user revocation scheme in mobile social networks. In Proceedings of the 2011 IEEE Global Telecommunications Conference-GLOBECOM 2011, Houston, TX, USA, 5–9 December 2011; pp. 1–5.
42. Yu, G.; Zha, X.; Wang, X.; Ni, W.; Yu, K.; Yu, P.; Zhang, J.A.; Liu, R.P.; Guo, Y.J. Enabling attribute revocation for fine-grained access control in blockchain-IoT systems. *IEEE Trans. Eng. Manag.* **2020**, *67*, 1213–1230. [CrossRef]
43. Waters, B. Ciphertext-policy attribute-based encryption: An expressive, efficient, and provably secure realization. In Proceedings of the Public Key Cryptography–PKC 2011: 14th International Conference on Practice and Theory in Public Key Cryptography, Taormina, Italy, 6–9 March 2011; Springer: Berlin/Heidelberg, Germany, 2011; pp. 53–70.
44. Nakamoto, S. *Bitcoin: A Peer-to-Peer Electronic Cash System*; Bitcoin.org 2008. Available online: https://bitcoin.org/bitcoin.pdf (accessed on 18 May 2023).
45. Ethereum Foundation. Ethereum White Paper. Available online: https://ethereum.org/en/whitepaper/ (accessed on 18 August 2022).
46. Androulaki, E.; Barger, A.; Bortnikov, V.; Cachin, C.; Christidis, K.; De Caro, A.; Enyeart, D.; Ferris, C.; Laventman, G.; Manevich, Y.; et al. Hyperledger fabric: A distributed operating system for permissioned blockchains. In Proceedings of the Thirteenth EuroSys conference, Porto, Portugal, 23–26 April 2018; pp. 1–15.
47. Lou, X.; Zhang, T.; Jiang, J.; Zhang, Y. A survey of microarchitectural side-channel vulnerabilities, attacks, and defenses in cryptography. *ACM Comput. Surv. (CSUR)* **2021**, *54*, 1–37. [CrossRef]
48. TCG. *Trusted Platform Module Library Part 1: Architecture*; TCG: Beaverton, OR, USA, 2019. Available online: https://trustedcomputinggroup.org/wp-content/uploads/TPM-Rev-2.0-Part-1-Architecture-01.16.pdf (accessed on 26 June 2023).
49. Xu, T.; Wendt, J.B.; Potkonjak, M. Security of IoT systems: Design challenges and opportunities. In Proceedings of the 2014 IEEE/ACM International Conference on Computer-Aided Design (ICCAD), San Jose, CA, USA, 3–6 November 2014; pp. 417–423. [CrossRef]
50. Attrapadung, N.; Imai, H. Attribute-based encryption supporting direct/indirect revocation modes. In Proceedings of the IMA International Conference on Cryptography and Coding, Cirencester, UK, 15–17 December 2009; Springer: Berlin/Heidelberg, Germany, 2011; pp. 278–300.
51. Allavena, A.; Demers, A.; Hopcroft, J.E. Correctness of a gossip based membership protocol. In Proceedings of the Twenty-Fourth Annual ACM Symposium on Principles of Distributed Computing, Las Vegas, NV, USA, 17–20 July 2005; pp. 292–301.
52. Raspberry Pi Foundation. *Raspberry Pi 4B*; Cambridge, UK. Available online: https://www.raspberrypi.com/products/raspberry-pi-4-model-b/ (accessed on 18 May 2023).
53. Infineon Technologies AG. *IRIDIUM SLI 9670 TPM2.0*; Munich, Germany. Available online: https://www.infineon.com/cms/en/product/evaluation-boards/iridium-sli-9670-tpm2.0/ (accessed on 18 May 2023).
54. Canonical Ltd. Open Source. *Ubuntu 20.04*. Available online: https://old-releases.ubuntu.com/releases/20.04/ (accessed on 18 May 2023).
55. Zeutro, L. *OpenABE*; GitHub, Inc.: San Francisco, CA, USA, 2018. Available online: https://github.com/zeutro/openabe (accessed on 18 May 2023).
56. Lenovo Group Limited. *ThinkPad E495*; Hong Kong, China. Available online: https://www.lenovo.com/de/de/p/laptops/thinkpad/thinkpade/e495/22tp2tee495 (accessed on 18 May 2023).
57. Berkovits, S.; Chokhani, S.; Furlong, J.A.; Geiter, J.A.; Guild, J.C. *Public Key Infrastructure Study*; Technical Report; National Inst of Standards and Technology: Gaithersburg, MD, USA, 1994.
58. *IEEE Std 802.1AR-2018*; IEEE Standard for Local and Metropolitan Area Networks-Secure Device Identity. (Revision of IEEE Std 802.1AR-2009). New York, NY, USA, 2018; pp. 1–73. [CrossRef]
59. Howard, M.; Lipner, S. *The Security Development Lifecycle*; Microsoft Press: Redmond, WA, USA, 2006; Volume 8.
60. Finne, T. The information security chain in a company. *Comput. Secur.* **1996**, *15*, 297–316. [CrossRef]
61. Roh, Y.; Heo, G.; Whang, S.E. A Survey on Data Collection for Machine Learning: A Big Data-AI Integration Perspective. *IEEE Trans. Knowl. Data Eng.* **2021**, *33*, 1328–1347. [CrossRef]
62. Qiao, Z.; Liang, S.; Davis, S.; Jiang, H. Survey of attribute based encryption. In Proceedings of the 15th IEEE/ACIS International Conference on Software Engineering, Artificial Intelligence, Networking and Parallel/Distributed Computing (SNPD), Las Vegas, NV, USA, 30 June–2 July 2014; pp. 1–6. [CrossRef]