Application of Attribute-Based Encryption in Military Internet of Things Environment

Łukasz Pióro *, Jakub Sychowiec, Krzysztof Kanciak, Zbigniew Zieliński

Posted Date: 30 July 2024

doi: 10.20944/preprints202407.2360.v1

Keywords: Internet of Things; Blockchain; Attribute-Based Encryption

Article

Application of Attribute-Based Encryption in Military Internet of Things Environment

Łukasz Pióro[1,†,‡]*, Jakub Sychowiec [1,†,‡], Krzysztof Kanciak [1,†,‡] and Zbigniew Zieliński [1,†,‡]

[1] Military University of Technology, Warsaw, Poland.
* Correspondence: krzysztof.kanciak@wat.edu.pl; lukasz.pioro@wat.edu.pl; Tel.: +48-608-687-159 (F.L.)
† Current address: Military University of Technology, gen. Sylwestra Kaliskiego 2, 00-908 Warsaw, Poland.
‡ These authors contributed equally to this work.

Abstract: Military Internet of Things (MIoT) has emerged as a new research area in military intelligence. The MIoT frequently has to constitute a federation-capable IoT environment when the military needs to interact with other institutions and organizations or carry out joint missions as part of a coalition such as in NATO. One of the main challenges of deploying MIoT in such an environment is to acquire, analyze, and merge huge amounts of data from many different IoT devices and disseminate it in a secure, reliable, and context-dependent manner. In this work, we present a comprehensive solution for secure data dissemination based on the integration of Hyperledger Fabric's distributed registry technology, Apache Kafka message broker, and data processing microservices implemented using the Kafka Streams API library and applying attribute-based encryption (ABE), a cryptographic approach that facilitates controlling access to information based on data and user attributes rather than traditional methods that rely on specific keys. In the work, we built an experimental setup, implemented an access control scheme designed for mobile components, and performed an evaluation of its performance. Performance tests have confirmed the suitability of the developed cryptographic method for attribute-based data access control in MIoT systems, considering the federations of military networks. The results indicate that the proposed cryptographic scheme is viable for the number of attributes typically assumed to be used in battlefield networks, offering a good trade-off between security and performance for modern cryptographic applications.

Keywords: Internet of Things; Blockchain; Attribute-Based Encryption.

1. Introduction

In the era of rapid technological advancement, the intersection of military operations and Internet of Things (IoT) technologies has established a new research field, the Military Internet of Things (MIoT). Connected devices, ranging from unmanned vehicles to smart surveillance systems, are now integral to military operations. This continuous technological development is reshaping the landscape of modern conflicts, presenting new opportunities and challenges for armed forces worldwide. Examples of military applications of IoT include smart logistics (transportation), smart bases, soldiers' health care, or Internet of Battle Things (IoBT) [1]. One of the main problems of MIoT deployment is acquiring, analyzing, and merging huge amounts of data from many different IoT devices and disseminating it in a secure, reliable, and context-dependent manner. The problem is compounded in these applications when the military has to interact with other institutions and organizations to form a federation, the purpose of which is to allow different cooperating parties to share some of the resources belonging to its various participants and to exchange acquired information from different sources. As military systems become increasingly interconnected, ensuring the confidentiality, integrity, and accessibility of sensitive data is paramount.

An increasing number of MIoT applications are expected to require the ability to operate in a federated environment. Examples include the formation of federations of NATO and non-NATO countries participating in joint missions (Federated Mission Networking) or the interaction of civilian and military services when providing humanitarian and disaster relief. Federations are often formed on an ad hoc basis, with the main goal of combining forces in a federated mission environment

 doi:10.20944/preprints202407.2360.v1

at any time, on short notice, and with optimization of the resources involved [2]. For example, we can cite the situation of a natural disaster such as an earthquake or flood that affects a smart city [3]. To coordinate effectively, it is necessary to have as complete and up-to-date a situational picture as possible. A system can be built using data from smart city devices, residents, and the military to provide a complete situational picture obtained in near real-time mode. In such cases, the establishment of a secure federated IoT environment is required, where different entities can have limited trust relationships and resources already owned by individual organizations are used. The use of federation has several significant advantages, which include simplification of processes, e.g. single authentication, cost reduction through simplification of communication infrastructure, minimization of data storage, efficient data exchange, sharing of resources, etc. When building a data exchange system for a federated IoT environment, we must consider several requirements, including the following. 1. Ensuring immediate interoperability (zero-day interoperability), that is, readiness for deployment when the need arises for a federated IoT environment. 2. Enable the authentication and authorization of IoT devices from various federation organizations. 3. Guarantee the security and reliability of data acquisition, flexible processing and distribution to potential recipients with resource-limited IoT devices. 4. Organizational decoupling: No organization can access the data and resources of another organization without its permission. 5. Flexible data management and enabling fine-grained access control to distributed data for different recipients.

One of the main challenges in a federated environment of separate IoT administrative domains is reliable device authentication and effective data access management, which are the basis for establishing trust relationships and secure communication between IoT devices belonging to different partners[4]. One potential solution to this challenge lies in applying Attribute-Based Encryption (ABE), a cryptographic approach that facilitates access control to information based on user attributes rather than traditional methods reliant on specific keys. Introduced by Sahai and Waters in 2005 [5], ABE departs from conventional encryption models by embedding access control policies directly in the ciphertext. Although numerous authors have further developed the approach, its practical applications remain relatively unexplored.

This work aims to develop an architecture framework for an experimental MIoT system to securely and reliably distribute various data types in a federation-capable MIoT environment. In the experimental system, device authentication is based on the identity of the IoT device (fingerprint). The system developed software IoT gateways to provide the verification process by integrating distributed registry technology (Hyperledger Fabric), a message broker (Apache Kafka), data processing microservices (using the Kafka Streams API library) and designed and implemented ABE cryptography data access control methods using lightweight elliptic curve cryptography. The primary objective was to evaluate the performance of the designed attribute-based cryptographic control method and its impact on the latency of data exchange between IoT end devices (sender and receiver). We see our contribution as follows. Firstly, we built the framework for secure and reliable data dissemination in a federated IoT environment, enabling IoT device authentication based on their identity (fingerprint), with the use of distributed registry technology (Hyperledger Fabric), a message broker (Apache Kafka), data processing microservices (using the Kafka Streams API library) and software IoT gateways providing the verification process. Second, we designed and implemented ABE cryptography data access control methods using lightweight elliptic curve cryptography and confirmed their suitability for a federated military IoT environment. Third, we built an experimental setup, implemented an access control scheme designed on mobile components (RPi), and performed an evaluation of its performance.

The remainder of the paper is organized as follows. Section 2 reviews related work on blockchain-supported Federated Networks and ABE. Section 3 briefly characterizes the MIoT architecture and presents the design and specification of requirements for a practical IoT health monitoring system. Section 4 explains mathematical foundations of ABE, along with more detailed description of processes in CP-ABE approach, used for experimental purposes. Section 5 presents the system's overall structure, intended to securely and reliably disseminate data across all command levels and facilitate interop-

 doi:10.20944/preprints202407.2360.v1

erability. We also highlight our solution for securing information from its inception to its disposal and implementing fine-grained access control mechanisms. In Section 6, we evaluate the feasibility of the proposed solutions by assessing the impact of the applied ABE method on the final delay in the dissemination of the message between the sender and the recipients. In the final section 7, we briefly summarize the results and identify directions for further work on developing the experimental system.

2. Related Works

Efforts are underway within NATO countries to integrate data exchange systems between coalition nations [6], resulting in the development of the Federated Mission Networking (FMN) concept. However, one of the significant challenges related to FMN and data dissemination systems is achieving zero-day interoperability. Jansen et al. [7] introduced an experimental MQTT-based environment involving four organizations, distributing data across two configurations. In a study by Suri et al. [8], an analysis and performance evaluation of eight data exchange systems (e.g.: RabbitMQ, Redis, DisService) within mobile tactical networks was conducted. De Rango et al. [9] propose a data exchange system for IoT devices based on the MQTT protocol, with data encrypted using elliptic curves. Additionally, Yang et al. [10] presented a system architecture designed for anonymized data exchange utilizing the Federation-as-a-Service cloud service model. Moreover, the literature abounds with efforts to integrate the IoT with blockchain technology, also known as distributed ledger technology. The work [11] delves into both the challenges and advantages of merging blockchain with IoT, emphasizing its implications for data security. Guo et al. [12] put forth a mechanism for authenticating IoT devices across various domains. Their approach leverages distributed ledgers operating in a master-slave configuration for seamless data exchange. Meanwhile, Xu et al. [13] introduced the DIoTA framework, which employs a private Hyperledger Fabric blockchain to safeguard the integrity of data processed by IoT devices. Furthermore, Khalid et al. [14] outlined an access control mechanism for IoT devices, utilizing the Ethereum public blockchain situated in the Fog Layer, complemented by a public key infrastructure based on elliptic curves. The work by Müller et al. [15] introduces the concept of Distributed Attribute-Based Encryption (DABE), where multiple parties can manage attributes and their corresponding secret keys, in contrast to the traditional centralized approach. This distributed model aligns well with the decentralized nature of IoT and blockchain systems. In [16] Jiang et al. propose a blockchain copyright protection scheme based on ABE. This approach aims to address security issues in copyright protection by leveraging the advantages of both blockchain and attribute-based encryption. Lu Ye et al. introduce an access control scheme that combines blockchain and CP-ABE in [17]. This scheme offers fine-grained attribute revocation, keyword search capabilities, and traceability of attribute private keys, enhancing security and flexibility in cloud health systems. Gondalia et al. [18] proposed an IoT-based healthcare monitoring system for war soldiers using machine learning. This system enables army control units to track soldiers' locations and monitor their health using GPS modules and wireless body area sensor networks (WBASNs). It incorporates various sensors to measure vital signs and environmental conditions. In [19] Sujitha et al. discuss an IoT-based healthcare monitoring and tracking system for soldiers. It emphasizes continuous remote monitoring of troops' health using IoT technology. Ashish and Kakali propose an edge computing-based secure health monitoring framework for electronic healthcare systems [19]. This framework incorporates ABE techniques to protect medical data and maintain secure monitoring of healthcare information. The research highlights the use of Ciphertext-Policy ABE (CP-ABE) for data security in IoT-based healthcare systems.

3. Military Internet of Things Environment

3.1. Environment Elements

MIoT environment represents a sophisticated combination of various interconnected devices and systems. Its primary objective is to enhance warfare capabilities, such as situational awareness,

 doi:10.20944/preprints202407.2360.v1

logistics, or medical care. The practical military environment comprises numerous devices operating at different levels [20].

At the tactical level, soldiers are equipped with various sensors and communication devices through programs such as the French Army's FELIN [21], the U.S. Army's Future Combat Systems [22] and NETT Warrior [23], or the Polish Army's Tytan.

Mechanized and motorized teams are provided with vehicles, including Infantry Fighting Vehicles (IFVs) or Armored Personnel Carriers (APCs). These vehicles have the potential to offer edge processing capabilities to the team, aligning with the imperative for minimizing the size of soldiers' equipment. Information from tactical units is transferred to the operational level, where data centers within command structures receive and process it. These data centers also receive data from other sources, such as specialized reconnaissance units or Unmanned Aerial Vehicles (UAVs). In cooperation with other commands and cloud support, they process the information to achieve situational awareness and formulate tasks for subordinate units (Figure 1).

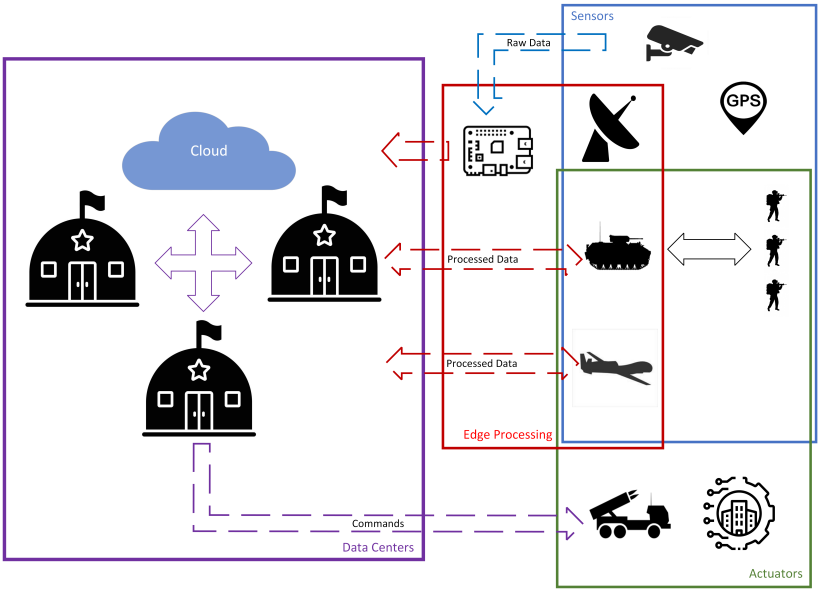

Figure 1. High-Level scheme of MIoT main components.

The architecture of the MIoT system is crucial in supporting military operations. It is intricately linked to the specific military scenario and the application domain. In this work, we adopt a general high-level architecture for military applications based on the features and requirements of various military scenarios. The general reference model of the MIoT four-layer architecture is shown in Figure 2.

Perception&Actuation layer (PAL) can be divided into data acquisition and Short-Range Wireless Communication (SRWC) parts and is related to the physical MIoT sensor/actuator nodes. Data acquisition collects information using RFID (Radio Frequency Identification), sensors, and other technologies. The SRWC concentrates the intercommunication of information on a small scale, for example, in the case of individual dismounted soldier sensors. The primary goal of this layer is to obtain real-time "sensed" data from the environment of each sensor node, including RFID tags/sensors, 5G warfighter phones, Global Positioning System (GPS), devices embedded with sensors, etc. Devices that function as actuators are responsible for carrying out physical actions based on commands received from IoT systems, converting digital signals into operational environments. As part of the MIoT, devices or platforms will include Unmanned Aerial/Ground Vehicles, mobile sensors, Unattended Ground Sensors, and individual wearable computer technologies (e.g., sensors that monitor vital soldiers' functions).

 doi:10.20944/preprints202407.2360.v1

Communication layer (CL) transmits the information received from the perception layer to any particular information processing system using existing military communication networks, including access networks (5G, Wi-Fi, etc.) or core networks (Internet/intranet). This layer comprises devices (routers, signal transceivers, etc.) responsible for reliable data transmission. In the military domain, considerable heterogeneity exists among the communication technologies that transmit data between entities. Additionally, non-civilian links such as VHF, HF, UHF, or satellite are used, resulting in network characteristics that strongly differ from conventional wireless communication.

Data Management Layer (DML) effectively processes massive amounts of data and enables intelligent decision-making based on network behavior. The core of data management is automated processing technology, which can help implement massive data extraction, classification, filtering, identification, and dissemination.

Intelligent Service Layer (ISL) consists of services comprising this layer, including data storage, computing, data search, system security, and application support. These services would typically be provided by middleware, usually based on the Service-oriented approach. In the MIoT, middleware deployment is essential to allow the abstraction of heterogeneous network technologies.

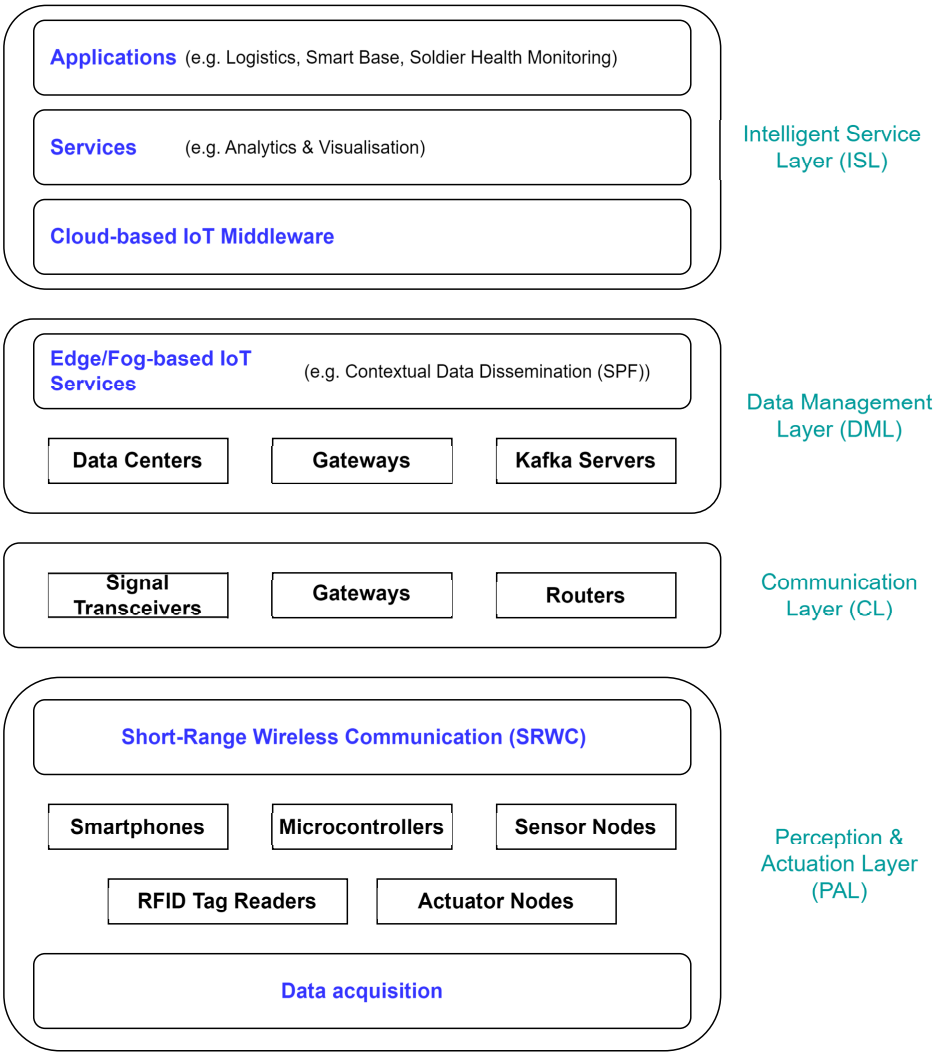

Figure 2. The MIoT layered architecture

3.2. System Requirements

This work presents the design and requirements specification for a practical soldier health-monitoring system leveraging IoT technology. While focusing on health monitoring, the proposed

 doi:10.20944/preprints202407.2360.v1

workflow and architecture can be adapted to various other use cases, such as situational awareness systems or battlefield management systems [24]. The system operates under the following assumptions:

1. Each soldier is equipped with various health-monitoring sensors, including but not limited to heartbeat sensors and thermometers.
2. Each sensor is seamlessly connected (paired) to an edge computing device integrated into the soldier's equipment. These edge devices can perform basic cryptographic operations to ensure data security and integrity.
3. A network infrastructure facilitates the communication between sensors, actuators, and central data centers.
4. Special services are responsible for processing incoming data from sensors, analyzing health metrics, and issuing commands to actuators as necessary.

These assumptions are fundamental for compliance with designed MIoT health monitoring systems discussed for example in [19] and [18]. Key features of these systems include:

1. Real-time tracking of soldier locations using GPS.
2. Monitoring of vital signs such as body temperature, heart rate, and oxygen levels.
3. Use of wireless body area sensor networks (WBASNs) for data collection.
4. Integration with IoT platforms for data transmission to command centers.
5. Potential incorporation of machine learning for data analysis and prediction.

These systems aim to improve soldier safety, enable rapid response to health emergencies, and enhance overall battlefield awareness for military commanders. By leveraging IoT technology, they provide a more comprehensive and real-time view of soldier health and location compared to traditional methods. The system should fulfill the following functional requirements regarding security:

1. **End-to-End Encryption**: All communication channels between sensors, edge devices, and data centers must be encrypted with a security level sufficient for IoT purposes, ensuring that data is protected from unauthorized access or tampering throughout its transmission, safeguarding sensitive health information.
2. **Authentication and Access Control**: Each device within the network must authenticate itself before participating in data exchange. Access control mechanisms should be enforced to restrict access based on roles and privileges, preventing unauthorized devices from accessing the network, while access control ensures that only authorized users or devices can interact with the system, reducing the risk of data breaches.
3. **Integrity Verification**: Data integrity checks should be implemented at each stage of data transmission to detect and prevent tampering or alteration of sensor data, through integrity verification mechanisms, such as cryptographic hash functions, ensuring that data remains unchanged during transit, maintaining its reliability and trustworthiness.
4. **Interoperability in Federated Environments**: Interoperability within systems, whether military or civilian, is essential for enhancing effectiveness and responsiveness. For instance, it allows health monitoring systems deployed by military units to share real-time health data with civilian healthcare providers, enabling timely medical interventions and resource allocation during joint operations or disaster response efforts. This aspect also gains significance in joint military operations conducted in federated environments involving allied forces. Interoperability extends beyond technical integration to encompass the harmonization of processes, standards, and protocols.
5. **Data-Centric Security**: With the proliferation of IoT devices and sensors worn by individuals, the volume and variety of health data generated have increased exponentially. Data-centric security focuses on protecting the data rather than solely relying on perimeter defenses, acknowledging that traditional security measures may be insufficient in dynamic and distributed environments. Encryption ensures that data remains unintelligible to unauthorized entities during transmission

 doi:10.20944/preprints202407.2360.v1

over the network, mitigating the risk of interception or eavesdropping. Access controls enforce granular permissions, allowing only authorized personnel/systems to access/manipulate health data, reducing the likelihood of data disclosure or breaches.

4. Attribute-Based Encryption

Attribute-based encryption (ABE) is a public-key encryption paradigm that provides fine-grained access control. In ABE, both the user's secret key and the ciphertext are associated with attributes that determine the ability to decrypt the ciphertext. It achieves this through the use of pairing-friendly curves and bilinear maps. Both KP-ABE and CP-ABE offer different approaches to policy enforcement, catering to various application requirements in secure data sharing.

There are two main types of ABE: Key-Policy Attribute-Based Encryption (KP-ABE) and Ciphertext-Policy Attribute-Based Encryption (CP-ABE). In KP-ABE, the ciphertext is associated with a set of attributes, and the user's secret key is associated with an access policy. The key policy (KP-ABE) determines which ciphertexts the user can decrypt. In contrast, in CP-ABE, the user's secret key is associated with a set of attributes, and the ciphertext is associated with an access policy. In the scope of this study, CP-ABE is used for the reasons specified in 5.3.

4.1. Bilinear Pairings

Attribute-Based Encryption schemes often rely on bilinear pairings on elliptic curves, known as pairing-friendly curves. Let $\mathbb{G}_1$ and $\mathbb{G}_2$ be cyclic groups of prime order p, and let $e : \mathbb{G}_1 \times \mathbb{G}_1 \to \mathbb{G}_2$ be a bilinear map. The properties of the bilinear map are the following:

- **Bilinearity:** $e(aP, bQ) = e(P, Q)^{ab}$ for all $P, Q \in \mathbb{G}_1$ and $a, b \in \mathbb{Z}_p$.
- **Non-degeneracy:** $e(g, g) \neq 1$ if g is a generator of $\mathbb{G}_1$.
- **Computability:** There exists an efficient algorithm to compute $e(P, Q)$ for all $P, Q \in \mathbb{G}_1$.

4.2. ABE Procedures

The CP-ABE scheme can be formally split into following procedures:

1. **Setup**(λ): This procedure initializes the system.
   (a) Choose bilinear groups $\mathbb{G}$ and $\mathbb{G}_T$ of prime order p. These groups are fundamental to cryptographic operations.
   (b) Select a generator $g \in \mathbb{G}$. This will be used to generate other group elements.
   (c) Choose random exponents $\alpha, \beta \in \mathbb{Z}_p$. These serve as the core secrets of the system.
   (d) Compute the public parameters:
      - $h = g^{\beta}$: This hides β while allowing its use in computations.
      - $f = g^{1/\beta}$: This is used in encryption and decryption.
      - $Y = e(g, g)^{\alpha}$: This is a pairing operation that hides α.
   (e) Set $\mathsf{PP} = (\mathbb{G}, \mathbb{G}_T, p, g, h, f, Y)$ as the public parameters.
   (f) Set $\mathsf{MSK} = (\beta, g^{\alpha})$ as the master secret key.
   (g) Output PP and MSK. PP is made public, while MSK is kept secret.
2. **KeyGen**(MSK, S): This procedure generates a secret key for an attribute set S.
   (a) Choose a random $r \in \mathbb{Z}_p$. This randomizes the key for security.
   (b) Compute $D = g^{(\alpha+r)/\beta}$. This embeds the master secret into the key.
   (c) For each attribute $i \in S$:
      i. Choose a random $r_i \in \mathbb{Z}_p$. This further randomizes each attribute component.
      ii. Compute $D_i = g^{r} \cdot H(i)^{r_i}$. $H(i)$ is a hash function mapping attributes to group elements.
      iii. Compute $D'_i = g^{r_i}$. This allows for cancellation in decryption.
   (d) Set $\mathsf{SK}_S = (D, \{(D_i, D'_i)\}_{i \in S})$ as the secret key for attribute set S.
   (e) Output SK_S.
3. **Encrypt**(PP, m, $\mathcal{A}$): This procedure encrypts a message m under an access policy $\mathcal{A}$.
   (a) Choose a random $s \in \mathbb{Z}_p$. This randomizes the encryption.
   (b) Compute $\tilde{C} = m \cdot Y^{s} = m \cdot e(g, g)^{\alpha s}$. This hides the message with the master secret.

 doi:10.20944/preprints202407.2360.v1

(c) Compute $C = h^s$. This is used in decryption to recover the message.
(d) For each attribute i in the access structure $\mathcal{A}$:

i. Choose a random $s_i \in \mathbb{Z}_p$. This randomizes each attribute in the policy.
ii. Compute $C_i = g^{s_i}$.
iii. Compute $C_i' = (g^s \cdot H(i)^{-s_i})^{1/\beta}$. This embeds the policy into the ciphertext.

(e) Set $\mathsf{CT} = (\mathcal{A}, \tilde{C}, C, \{(C_i, C_i')\}_{i \in \mathcal{A}})$ as the ciphertext.
(f) Output CT.

4. **Decrypt**($\mathsf{PP}, \mathsf{CT}, \mathsf{SK}_S$): This procedure attempts to decrypt a ciphertext using a secret key.

(a) First, check if the attribute set S satisfies the access policy $\mathcal{A}$. If not, output $\perp$ (decryption failure).
(b) If S satisfies $\mathcal{A}$:

i. Compute $e(g,g)^{\alpha s}$ using CT and SK_S:
- This involves pairing operations and computations based on the satisfying set of attributes.
- The computation cancels out randomizing factors, leaving only $e(g,g)^{\alpha s}$.

ii. Recover the message: $m = \tilde{C}/e(g,g)^{\alpha s}$.
iii. Output the decrypted message m.

In the decryption process of Ciphertext-Policy Attribute-Based Encryption (CP-ABE), recovering $e(g,g)^{\alpha s}$ is a crucial step. The decryption algorithm uses the ciphertext CT and the secret key SK_S associated with an attribute set S that satisfies the access policy $\mathcal{A}$. The decryption process involves pairing operations and computations using these components. The goal is to cancel out all the randomizing factors, leaving only $e(g,g)^{\alpha s}$. In simplified version:

$$\frac{e(C,D)}{\prod_{i \in I} \frac{e(C_i,D_i)}{e(C_i',D_i')}} = \frac{e(h^s, g^{(\alpha+r)/\beta})}{e(g,g)^{rs}} = \frac{e(g^{\beta s}, g^{(\alpha+r)/\beta})}{e(g,g)^{rs}} = \frac{e(g,g)^{\alpha s} \cdot e(g,g)^{rs}}{e(g,g)^{rs}} = e(g,g)^{\alpha s}$$

where I is the set of attributes used in the decryption and $\frac{e(C_i,D_i)}{e(C_i',D_i')}$ is a term designed to equal $e(g,g)^{rs}$.

The exact computation depends on the specific CP-ABE scheme used, but the general idea is to use the properties of bilinear pairings and the structure of the keys and ciphertext to isolate $e(g,g)^{\alpha s}$.

5. Data Exchange System

The following headings provide an overview of our multilayered experimental system for IoT-based environments and explain the rationale for implementing a Data-Centric Security approach within our architecture.

5.1. Experimental Environment Architecture

Figure 3 depicts the overall structure of the system, which is intended to securely and reliably disseminate data across all command levels and facilitate interoperability among different entities, including sensors, actuators, and data centers, from various organizations. The environment (Figure 3) illustrates a federation of two organizations (Org1, Org2) and consists of several layers that perform the following functions:

- **The Publisher's Layer** represents authenticated and authorized entities (sensors and actuators) that produce and secure messages through the sealing process. The device's fingerprint (identity) is the key used for the sealing process.
- **Subscribers Layer** comprises authenticated and authorized entities that read available data from the Kafka Cluster layer.
- **Kafka Cluster Layer** is compromised of Apache Kafka message brokers that acquire, merge, store, and replicate data generated from the Publishers layer (producers) and make it available to the Subscribers layer (consumers). Apache Kafka operates on a producer-broker-consumer (publish-subscribe) model, facilitating the classification of messages based on their respective

 doi:10.20944/preprints202407.2360.v1

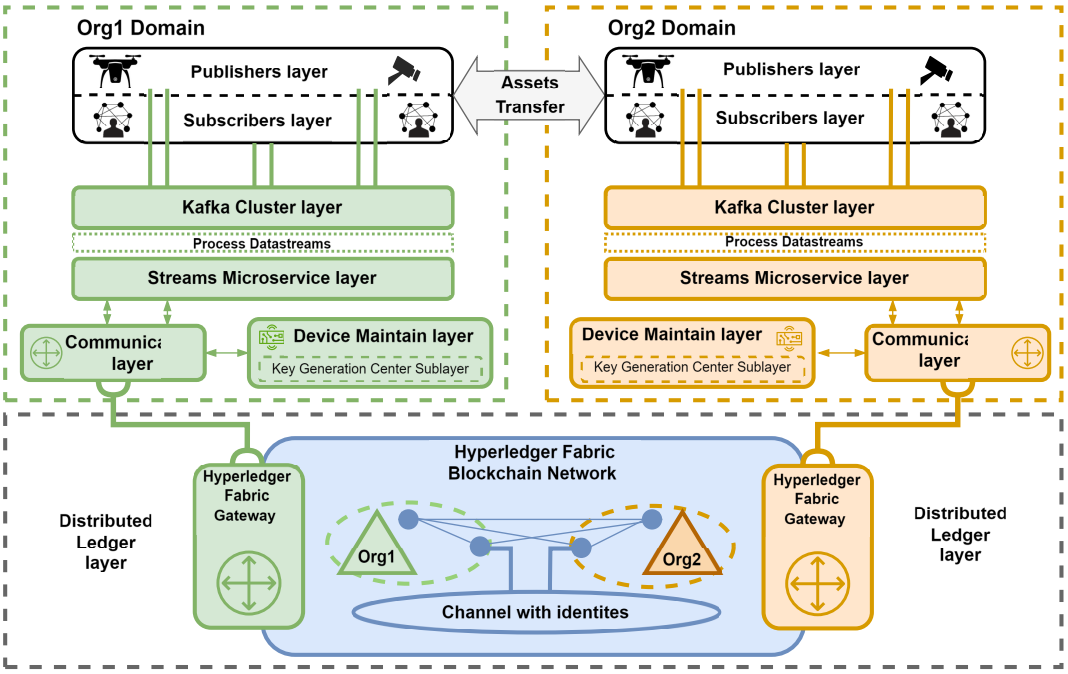

Figure 3. General overview of the experimental environment.

topics. The inherent synchronization mechanisms and distributed data replication among brokers ensure the continuous availability and reliability of data records. Furthermore, Kafka's serialization and compression techniques (e.g., lz4, gzip) enable the system to remain agnostic to data formats and network protocols, thereby ensuring compatibility and robustness in heterogeneous environments.

- **Streams Microservice Layer** is primarily utilized to verify sealed messages. Additionally, it can be used to analyze, group, and share messages related to relevant entities and enrich them (e.g., by detecting objects during image processing). The system leverages the built-in primitives of the Kafka Cluster layer, such as failover and fault tolerance. Additionally, it employs a semantic guarantee pattern ensuring that each record (message) is processed exactly once end-to-end. Consequently, even in the event of a stream processor (microservice) failure, records are neither lost nor processed multiple times. In the proposed system, it is also utilized for ABE re-encryption.
- **Device Maintain Layer** manages (e.g., define, register, retire) device identity image stored in the distributed ledger. In the proposed system, it is also responsible for ABE attribute and device management.
- **Communication Layer** enables the Streams Microservice and the Device Maintain layer to communicate with the Distributed Ledger layer via a hardware-software IoT gateway. Moreover, a dynamic mode is proposed for the connection profile. This profile utilizes the ledger nodes' built-in mechanism to continuously detect changes in network topology. Consequently, microservices will be able to operate reliably, even in the event of some node failures.
- **The Distributed Ledger Layer** redundantly stores the identities of devices belonging to organizations participating in the federation. A permissioned blockchain that employs the Practical Byzantine Fault Tolerance (PBFT) consensus protocol is proposed. In protocols of this nature, all participants must be mutually known, necessitating the use of a public key infrastructure (certificates) for identity verification. The execution of complex business logic, such as device registry, is facilitated by invoking multilingual chaincode (Go, Java, Node.js). Chaincode implements a collection of smart contracts (transaction steps) and defines an endorsement policy, specifying which organizations must authorize a transaction.

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 30 July 2024 doi:10.20944/preprints202407.2360.v1

The article by Zieliski et al. [25] discusses the rationale behind the use of components within the system layers, which is not covered in this publication. The following section outlines the flow of messages and the specific actions taken within our system, with Figure 4 providing a detailed illustration of our experimental system.

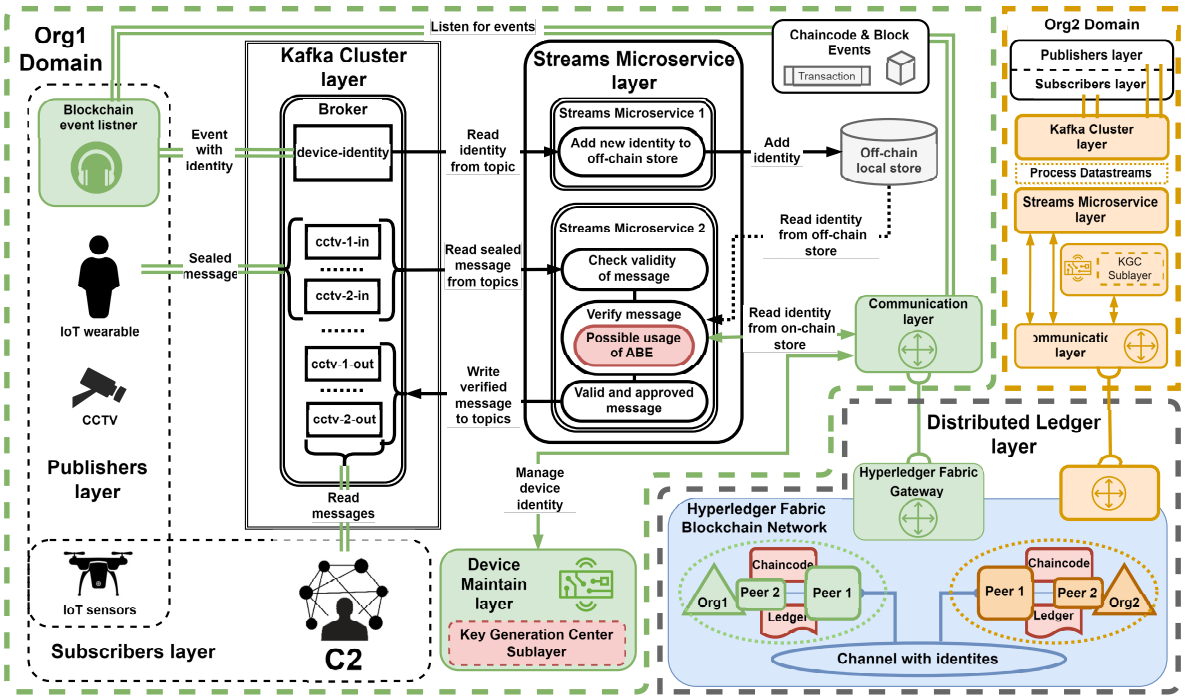

Figure 4. Detailed overview of the experimental environment.

Within our system, entities representing the Publisher's Layer secure their messages by sealing (tagging) them with registered identity. These sealed messages are then sent using the available communication protocol and medium to the Kafka Cluster layer, where they will be stored on a specific topic, such as "cctv-1-in". The next step involves the Streams Microservice layer, which reads messages from brokers belonging to the Kafka Cluster layer to verify sealed messages.

During verification, the microservice queries the Hyperledger Fabric ledger for an image of the device identity through the IoT gateway (Communication Layer). This query aims to compare the identity extracted from the message with the one stored in the Distributed Ledger layer. Once a message is verified and approved, it is written again on the Kafka Cluster layer within a dedicated topic (e.g., cctv-1-out) and provided to entities representing the Subscribers layer.

5.2. Authorization Challenges

The Apache Kafka technology has built-in components and mechanisms to define and handle entity (device) authorization using Access Control Lists (ACLs). In a nutshell, ACLs specify which entities can access a specified resource and the operations they can perform on that resource. One of the specific types of resource is a Kafka topic, which organizes (stores) messages produced by the Producers layer. The ACL operation depends on a resource type, and in the context of our solution, the READ and WRITE operation assigned to the entity (principal) for the topic is relevant.

Establishing a separate principal for each entity (device) and assigning only the necessary ACLs streamlines debugging and auditing, as it identifies which entity is performing each operation. For instance, within an environment with three devices, should be created three principals: device_job_01, device_job_02, and device_job_03. Afterward, for each principal, only the essential operations should be allowed, and each should be used with its designated principal. Kafka ACLs are defined in the general format of *"Principal P is [Allowed/Denied] Operation O From Host H On Resources matching Resource/Pattern RP"*. An example ACL rule definition is presented below:

 doi:10.20944/preprints202407.2360.v1


```
kafka-acls --bootstrap-server localhost:9092
    --command-config configs.conf  --add
    --allow-principal User:uav_1 --allow-host
    192.168.1.13 --operation write  --topic cctv-1-in
```

It means that the principal (user) named uav_1 is allowed to perform a WRITE operation on the topic cctv-1-in from IP address 192.168.1.13. In this example, a basic plain text principal was used. For real-world use case scenarios, protocols such as the SASL security protocol with GSSAPI (Kerberos) or TLS are more likely to be utilized. This implies that different security (authentication) protocols require different formats of the principal:

```
--deny-principal User: CN=uav_2, OU=Org2, O=None, C=None
```

Moreover, large Kafka Cluster topologies with numerous Kafka partitions or many publishing/subscribing entities can face significant challenges in managing entity authorization. Implementing a more complex authorization hierarchy in the Kafka Cluster layer is possible, but it also comes with operational overhead that requires the following:

- deploy an external server that associates users with roles and/or groups (e.g., LDAP server);
- define detailed ACL by specifying permission type for group, role, and user; (ACL stored in Kafka cluster layer);
- integrates a custom implementation of Authorizer with the user association server.

5.3. Attribute-Based Encryption Application

One of the possible solutions to the mentioned challenges is the application of ABE in our experimental environment. This can provide more fine-grained access control than the Kafka Cluster layer. Using ABE ACLs in our solution enables dynamic authorization at the individual message level. Additionally, this can relieve the Kafka Cluster layer from the burden of the authorization process, allowing its internal resources to be released, and the Streams Microservice layer can handle access control. In addition, operational costs are low and the ABE can be implemented in environments of any scale that involve multiple participating organizations. Finally, entities within organizations (such as the Publishers/Subscribers layer) can maintain the ability to move freely between Kafka Cluster layers in different organizations.

Recent ABE schemes are tailored to address specific requirements, showing variability in their capabilities, as highlighted in [26]. In the scope of this study, we opted for the scheme introduced by [27], mainly due to its provision of:

1. Unbounded attribute sets and policies: The algorithms can handle attribute sets and policies of any size without predetermined limits.
2. Support for negation and multi-use of attributes: This allows for more complex and expressive access control policies.
3. Fast decryption: The schemes are designed to perform decryption operations quickly, enhancing overall system performance.
4. Full security under standard assumptions: The algorithms provide strong security guarantees based on well-established cryptographic assumptions.

This choice aligns with the architecture of Kafka's Topics, for which attributes represented in a key:value pattern are easily processed. The necessity for repeated attributes in this pattern within Kafka's topics' architecture underscores the compatibility and appropriateness of the selected ABE scheme for the application of our system for the soldiers' health monitoring system.

 doi:10.20944/preprints202407.2360.v1

5.4. Design Proposition

5.4.1. General Overview

The data-centric security paradigm necessitates secure handling of information from its inception to its disposal, coupled with the implementation of fine-grained access control mechanisms. While ABE addresses these imperatives, it imposes substantial computational and bandwidth requirements. This study advocates a paradigm shift, since the sensors and edge processing layers predominantly comprise devices with limited computational capabilities. Specifically, we propose a scheme in which the responsibility for ABE encryption is delegated to the organizational infrastructure, presumed to possess the requisite cryptographic capabilities. Concurrently, sensors and their paired edge processing devices are envisaged to execute less computationally intensive cryptographic operations, such as AES encryption, SHA256 hashing, and HMAC-SHA256.

In the proposed framework (Figure 5), sensors employ symmetric encryption to encapsulate the generated data. Although the choice of encryption algorithm is not the focal point of this study, it prompts discussion on the potential applicability of lightweight cryptography. Devices with considerable computational prowess, such as Raspberry Pi or smartphones, predominantly utilize AES encryption. However, encryption and HMAC keys are derived from a sensor-specific fingerprint using a predefined key-derivation function. The sensor divides the generated message into two distinct components:

- **Encrypted Data Segment**: Encrypted with key derived from device fingerprint, consists of: 1. Encrypted Message; 2. Encrypted ABE policy; 3. Unencrypted Session GUID.
- **Preamble**: Used to authenticate message, consists of: 1. HMAC of Encrypted Data Segment (with key derived from fingerprint); 2. Device GUID; 3. Session GUID.

Subsequently, these components are published in Kafka input topics. Incoming messages undergo processing through the following sequence of operations within Kafka microservices: 1. Retrieval of the sensor fingerprint from Hyperledger using the provided device GUID; 2. key derivation for encryption and HMAC; 3. HMAC verification; 4. message decryption; 5. Attribute-based re-encryption based on the policy encapsulated within the main segment; 6. publication of the ABE-encrypted message on the topic specified in the policy. Microservices require at least one of the attributes to be a topic attribute. This ensures that topics associated with data recipients remain unencumbered.

The aforementioned mechanism is exclusively employed for streaming-type connections to distribute the session key associated with the Session GUID. Subsequent messages comprise solely the message encrypted with the session key and the Session GUID.

5.4.2. Attribute-Based Encryption Message Flow

Consider the transmission scenario of sending a report from a sensor:

1. The sensor generates data; for example, a smartphone equipped with a heart rate sensor detects an anomaly in heart activity. This information is appended with an attribute logic sentence and encrypted using the device fingerprint. Let the logic sentence be MEDICAL and GRID28B, where GRID28B represents the sector number obtained from the location sensor, also associated with this smartphone. GUIDs and HMAC are further appended as delineated in the system overview.
2. The sensor sends the data to a Kafka input topic, where processing is executed as described in the system overview.
3. Devices subscribed to the MEDICAL topic receive the message. If they satisfy the attached attribute logic sentence, they decrypt the message using their current keys at the time of data transmission. The Kafka microservice preliminarily verifies the authenticity of the message.

 doi:10.20944/preprints202407.2360.v1

Figure 5. Diagram of data flow in proposed system.

5.4.3. Attribute-Based Encryption System Setup

The initialization of the ABE system encompasses the following sequential steps (Figure 6) and is compliant with the Secure Onboarding process described in [28]:

1. Establishment of ABE system parameters, including the master secret key.
2. Pairing data recipients possessing limited computational capabilities with more robust devices by sharing a common symmetric key for communication, treating them as a unified entity.
3. Loading ABE system parameters onto the designated devices intended for system operation.
4. Associate private keys with device identifiers ID on devices designated for data reception. An identifier is any piece of information used to uniquely distinguish a device, such as a Device GUID generated in this step or a combination of type and number, e.g., UAV-1. These keys

 doi:10.20944/preprints202407.2360.v1

enable the distribution of attribute keys for specific devices when granting new permissions or authenticating attribute requests. This process is recorded in Hyperledger as Device Registration.
5. Loading private keys corresponding to granted attributes onto the respective devices. This action is documented in Hyperledger as attribute granting.

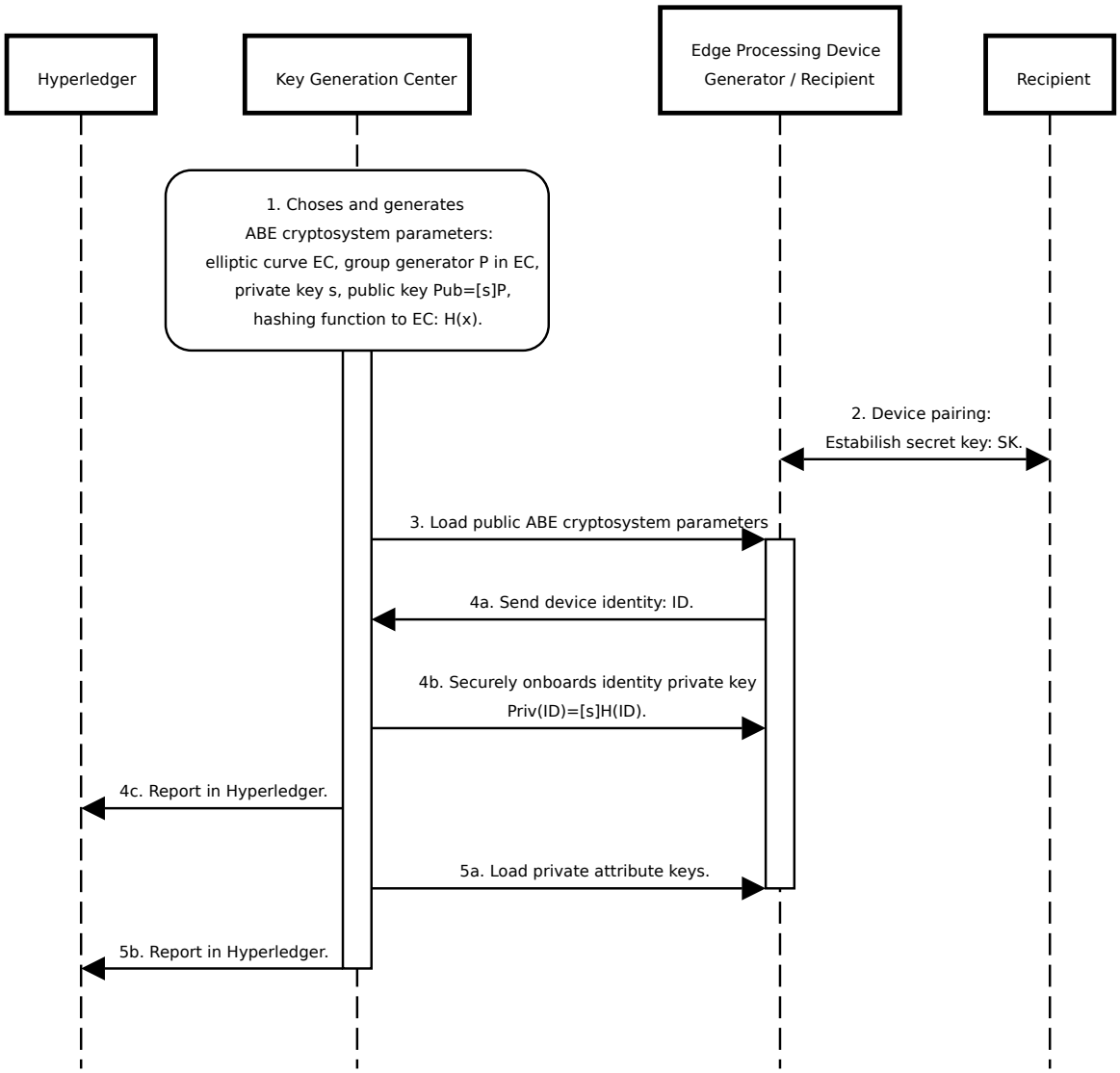

Figure 6. Sequence diagram illustrating the ABE system setup steps.

5.4.4. Attribute and Device Revocation

A significant challenge associated with ABE systems is related to the revocation of attributes or devices. While Public Key Infrastructure (PKI) solutions mitigate this issue via Certificate Revocation Lists, ABE lacks such certificates. Consequently, ABE commonly employs mechanisms such as timestamps for attribute or device management. However, the integration of Hyperledger allows for an alternative approach, associating attributes with the block number of the most recent revocation. Consider the following scenario (Figure 7):

1. A specific attribute, such as MEDICAL, is granted to any number of devices, with each grant action recorded in Hyperledger.
2. Upon revocation of the recipient attribute, this incident is documented in Hyperledger.

 doi:10.20944/preprints202407.2360.v1

3. New attributes associated with a revocation block number, e.g.,UAV-123, are issued to all other devices. Distribution occurs by encrypting new private keys with each device's identity and publishing them to the Kafka device management topic.
4. The microservice responsible for ABE re-encryption employs the new attribute public key.

If, instead of an attribute, a device is revoked from the system, KGC has to issue new keys for all attributes the device possesses to other recipients. Additionally, compromised data generators can be reported in Hyperledger to ignore messages originating from them. This can be facilitated through device GUID revocation. It is crucial to recognize that distributing new keys through a Kafka topic mitigates any potential data availability gaps. These keys become accessible concurrently with the encrypted data in the topics, ensuring seamless data access.

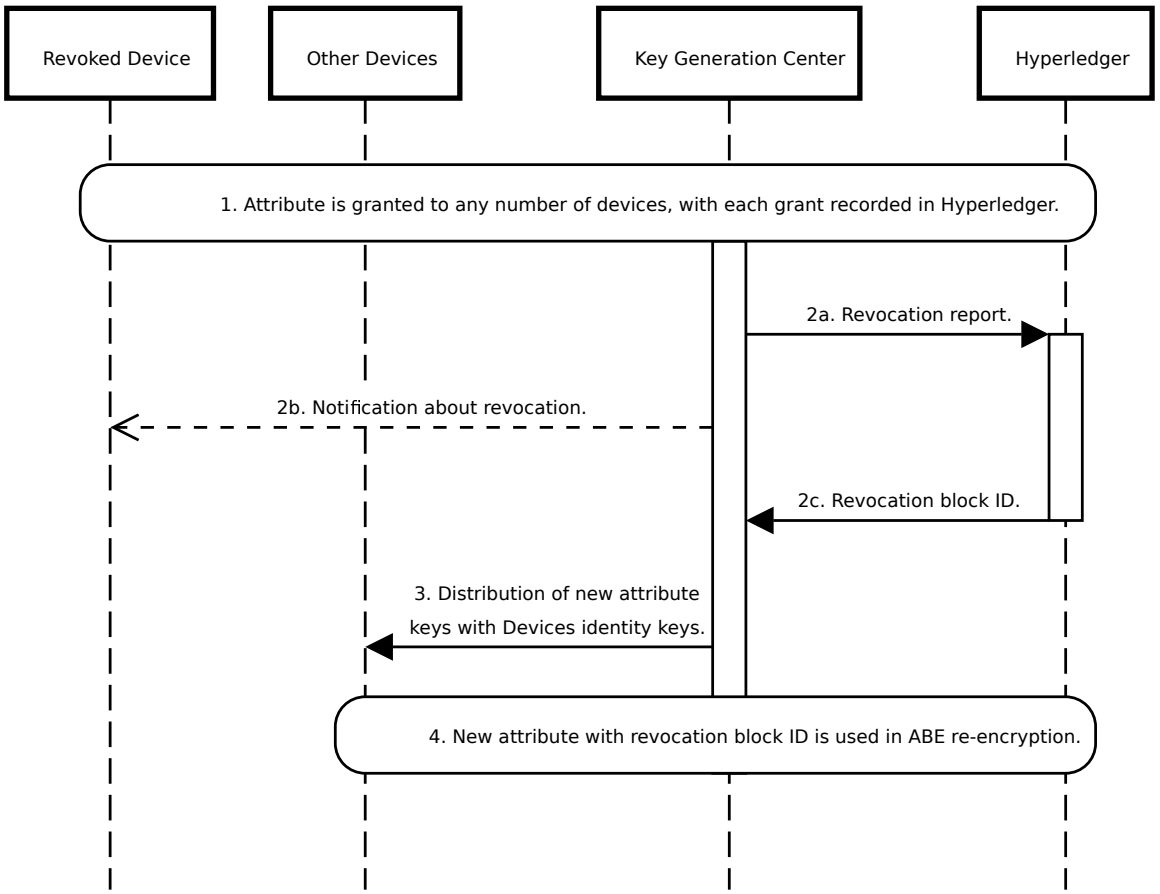

Figure 7. Sequence diagram illustrating the ABE attribute revocation steps.

5.4.5. Granting new permissions

Consider a scenario of granting new permissions (Figure 8):

1. The recipient device sends a request to KGC signing it with its identifier.
2. In KGC, after signature verification, a decision is made to grant a new key. It is reported on Hyperledger and sent back to the device, signed by KGC, and encrypted with the recipient identifier.
3. After signature verification and decryption, if an attribute is of topic type, the device subscribes to the new topic in the Kafka broker.

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 30 July 2024 doi:10.20944/preprints202407.2360.v1

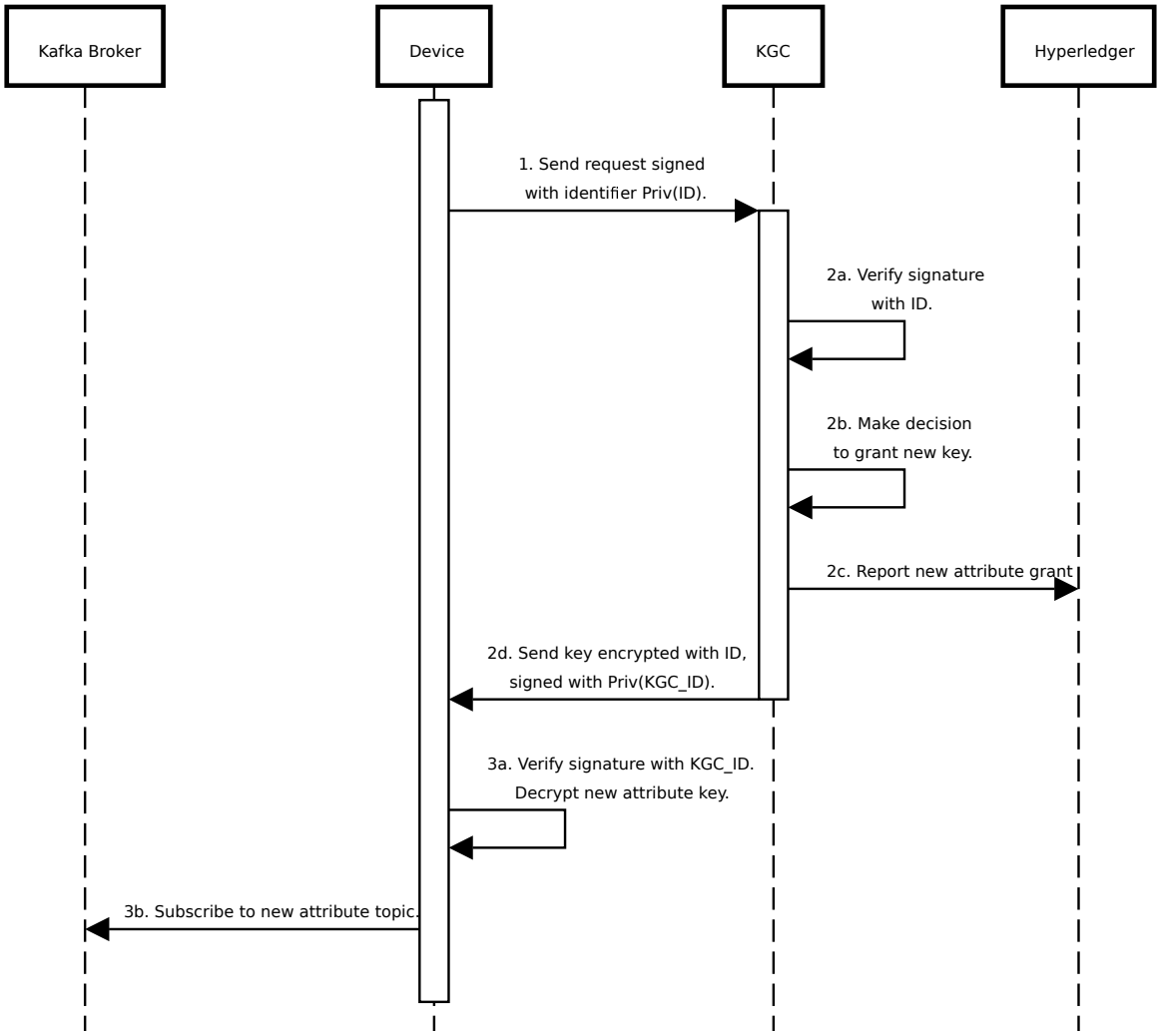

Figure 8. Sequence diagram of requesting new permissions.

6. Experimental Results

6.1. Experiments Objective

The primary objective of the experiments conducted on the described system is to ascertain its feasibility for deployment in real-world conditions. To investigate this hypothesis, the following metrics have been employed, providing a framework for evaluating the system's performance, security, and scalability in practical scenarios:

1. **Encryption and Decryption Time using ABE**: Evaluates the time required to encrypt and decrypt data using ABE.
2. **Latency in Data Transmission via Kafka and Device Verification in Hyperledger**: Measures the delay associated with transmitting data through the Kafka stream broker and the subsequent retrieval and verification of data within the Hyperledger framework.
3. **Key Generation Time**: Examines the time taken to generate new encryption keys when new keys are issued or existing keys are revoked.
4. **Size of the Resulting Ciphertexts**: Assesses the size of the encrypted data produced through the ABE processing.

 doi:10.20944/preprints202407.2360.v1

6.2. MIoT Context

In the experiments, efforts were made to replicate the operational conditions experienced by units of various levels on the battlefield. Given the assumptions and requirements outlined in 3.2, we assume that soldiers on the battlefield are equipped with medical monitors (data generators) paired with devices possessing sufficient computational power to aggregate signals from the soldiers' medical sensors and perform basic cryptographic operations (AES, HMAC) as described in 5.4, as well as decrypt ABE ciphertexts (for cases where the soldier is both the producer and consumer of data). For the purposes of the experiment, this device is represented by a Raspberry Pi 4 or 5. The organization that operates the medical system has its data center at the command post of the corresponding unit, which contains a Kafka broker, Hyperledger Fabric blockchain, and microservices. This data center is situated in a location such as a container mounted on a truck (e.g., Polish AWRS) and has sufficient space to accommodate larger devices. For the experiment, these devices are represented by the x86 platform, as well as clusters of smaller devices like the Raspberry Pi 5. The devices within the container are interconnected via wired connections, while wireless connections are used with other devices. Data recipients are soldiers on the battlefield, equipped with devices capable of performing ABE decryption, represented by Raspberry Pi 4/5, as well as other data centers, and command apparatuses represented by the x86 platform.

6.3. Implementation

The described environment and data exchange system have been realized in a simplified setup, comprising a data generator, a data recipient, and a Kafka broker. The Kafka broker is deployed within two Raspberry Pi 5 clusters as three instances (two instances on one of the devices and one on the other) with Hyperledger and ABE re-encryption microservice interconnected through Kafka Streams mechanisms. Hyperledger is deployed on x86 machine, while the re-encryption service is deployed interchangeably on Raspberry Pi 4, Raspberry 5 or x86 machine. The data generator produces and encrypts messages as described in the previous section, while the data recipient subscribes to the output topic within the Kafka broker. The data generator and the recipient are implemented as Go scripts. Data generator, re-encryption service, and data recipient scripts are available in [29]. For ABE, the CIRCL cryptographic library by Cloudflare [30], implementing [27], was integrated. Elliptic curve used is BLS12-381 described in [31] and [32], providing 126 bits of security [33]. BLS12-381 belongs to the Barreto-Lynn-Scott (BLS) family of curves, with the number 12 indicating its embedding degree and number 381 refering to the size of its prime field: a 381-bit prime. This curve operates over a prime field $\mathbb{F}_p$, where p is a 381-bit prime:

$$\begin{aligned} p = 0x1a0111ea397fe69a4b1ba7b6434bacd7 \\ 64774b84f38512bf6730d2a0f6b0f624 \\ 1eabfffeb153ffffb9feffffffffaaab \end{aligned}$$

and features a curve order q that is a 255-bit prime:

$$\begin{aligned} q = 0x73eda753299d7d483339d80809a1d805 \\ 53bda402fffe5bfeffffffff00000001. \end{aligned}$$

It defines three groups: $\mathbb{G}_1$, a subgroup of points on the curve $E(\mathbb{F}_p) : y^2 = x^3 + 4$; $\mathbb{G}_2$, a subgroup of points on the twist curve $E'(\mathbb{F}_{p^2}) : y^2 = x^3 + 4(u+1)$, where $\mathbb{F}_{p^2} = \mathbb{F}_p[u]/(u^2+1)$; and $\mathbb{G}_T$, the target group for pairing operations. This structure supports an efficiently computable bilinear pairing $e : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$. BLS12-381 is particularly useful in scenarios requiring pairing-based cryptography, offering a good trade-off between security and performance for modern cryptographic applications.

The logical scheme of the simplified setup is presented in 9.

 doi:10.20944/preprints202407.2360.v1

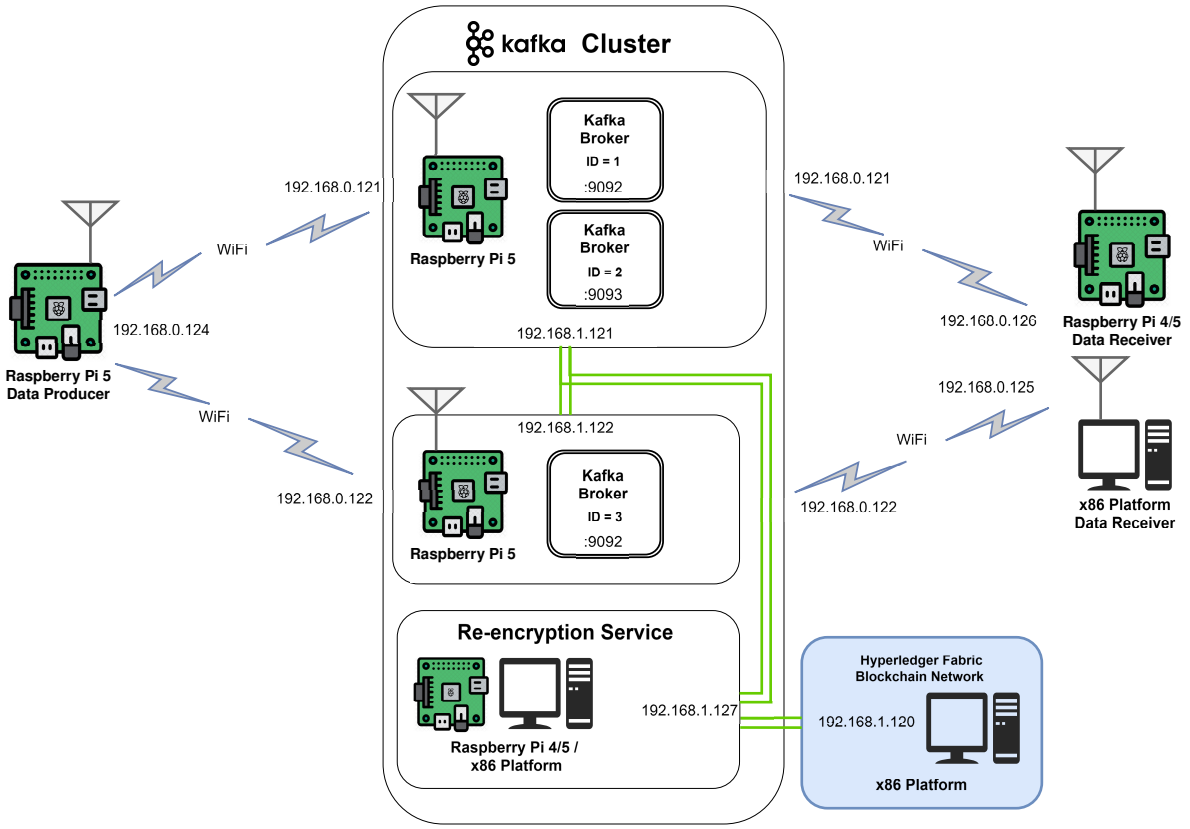

Figure 9. Scheme of experimental setup.

6.4. Performance Evaluation

The flow of messages from the generator to the recipient was emulated using the following methodology. The time taken for ABE encryption and decryption of 32-byte messages was measured in varying numbers of attributes, along with the length of the ciphertext for 1,000 messages. The results are presented in Table 1. Extreme values were excluded in each quantile order (p0.99, p0.95, and p0.9). The emulation was executed on a PC equipped with an x86 13th Gen Intel(R) Core(TM) i5-13600KF processor operating at 3.50 GHz. Furthermore, performance tests were conducted on a Raspberry Pi 4 and Raspberry Pi 5 using the same number of attributes as in the PC test. Both the data generator and the recipient were executed on the Raspbian OS as Golang scripts.

The results indicate that the proposed scheme is viable for the number of attributes typically assumed to be used in battlefield networks. For scenarios involving eight attributes, the processing time for encryption and decryption increases approximately by 100 ms, even when all messages are encrypted using ABE. Using session keys that incorporate ABE introduces only a minimal increase in the delay in connection establishment. The computational time of AES is assumed to be negligible.

Table 1. Simulation results on x86 processor.

		Abe Encrypt				Abe Decrypt		
Message length [bytes]	Number of attributes	Percentile 0.9	Percentile 0.95	Percentile 0.99	Ciphertext Length [bytes]	Percentile 0.9	Percentile 0.95	Percentile 0.99
32	1	35.88ms	35.85ms	35.70ms	2384	15.27ms	15.35ms	15.42ms
32	2	43.50ms	43.40ms	43.80ms	2744	18.49ms	18.12ms	17.87ms
32	3	46.78ms	46.64ms	46.27ms	3093	15.49ms	15.57ms	15.63ms
32	5	73.70ms	73.34ms	72.91ms	4567	20.42ms	20.20ms	20.26ms
32	8	83.43ms	83.72ms	83.20ms	4858	21.25ms	21.29ms	21.38ms

While there is a significant overhead associated with ciphertext size, this challenge has been effectively addressed through the adoption of hybrid cryptography. In this framework, Attribute-Based

 doi:10.20944/preprints202407.2360.v1

Encryption (ABE) is utilized primarily for session key encryption and distribution. For individual messages, ABE can be employed to encrypt only the symmetric key.

Nevertheless, ciphertexts of approximately 4KB can still present challenges for IoT networks that rely on wireless protocols. To transmit around 4000 bytes of ciphertext over networks with limited payload sizes, the data must be fragmented into multiple packets. This fragmentation not only introduces additional overhead but also complicates the communication process, leading to increased transmission times and higher energy consumption. For instance, the maximum payload size for LoRaWAN is 242 bytes with each packet consisting of additional 13 bytes for headers and other protocol-specific information[34]. In comparison, widely used cryptographic scheme RSA-3072, similarily to BLS12-381 believed to provide approximately 128 bits of security, generates ciphertexts of around 384 bytes.

The results presented in Table 2 and Table 3 indicate that ABE encryption may not be feasible for devices with limited computational capabilities. In contrast, ABE decryption requires considerably less time, which supports the strategy of managing ABE re-encryption within the Kafka microservice and executing only decryption on the recipient devices.

Table 2. Simulation results on Raspberry Pi 4.

		Abe Encrypt				Abe Decrypt		
Message length [bytes]	Number of attributes	Percentile 0.9	Percentile 0.95	Percentile 0.99	Ciphertext Length [bytes]	Percentile 0.9	Percentile 0.95	Percentile 0.99
32	1	736.44ms	736.41ms	736.34ms	2384	281.45ms	281.47ms	281.58ms
32	2	905.95ms	906.01ms	906.05ms	2744	291.47ms	291.42ms	291.37ms
32	3	1.0780s	1.0791s	1.0786s	3093	301.55ms	301.57ms	301.52ms
32	5	1.4257s	1.4246s	1.4251s	4567	321.41ms	321.49ms	321.44ms
32	8	1.9390s	1.9387s	1.9393s	4858	351.69ms	351.75ms	351.73ms

Table 3. Simulation results on Raspberry Pi 5.

		Abe Encrypt				Abe Decrypt		
Message length [bytes]	Number of attributes	Percentile 0.9	Percentile 0.95	Percentile 0.99	Ciphertext Length [bytes]	Percentile 0.9	Percentile 0.95	Percentile 0.99
32	1	444.75ms	443.71ms	443.14ms	2384	201.76ms	201.76ms	201.74ms
32	2	550.68ms	550.11ms	549.76ms	2744	213.71ms	213.74ms	213.73ms
32	3	657.90ms	657.52ms	656.59ms	3093	226.44ms	226.43ms	226.42ms
32	5	862.46s	861.99ms	862.10ms	4567	250.35ms	250.35ms	250.34ms
32	8	1.1826s	1.1817s	1.1814s	4858	287.00ms	287.00ms	286.99ms

In [25], it was demonstrated that the processing times for Kafka deployed in AWS averaged approximately 40 ms per message (with $IdentityCount = 100000$). This processing involved receiving messages on a topic, verifying the data generator in the blockchain, and forwarding the data to an output topic. In this experimental setup, average time of Kafka responding to requests was 17.3ms and average time of verifying the data generator in the blockchain was 31.3ms with $IdentityCount = 10000$, summing up to average time 48.6ms of additional processing of the packet between it appearing in the broker and receiving by consumer. Time of conducting other operations, such as calculation of HMAC and AES decryption, is negligible in comparison. The combined processing times with ABE re-encryption are detailed in Table 4 and Table 5.

It is important to note that, provided the Kafka broker, blockchain, and microservices possess adequate computational power and the network is not congested, the disadvantage of employing ABE for symmetric key distribution is the latency associated with establishing the streaming session, but does not impact the continuous processing of the data stream. The observed latency for establishing a streaming session is less than 0.5 seconds for 8 attributes on both Raspberry Pi 4 and 5.

In Table 6, the average times for new key generation as a function of the number of attributes are presented. The data indicate that delegating these computations to devices with constrained computational capabilities is impractical for large networks. Within the proposed system, while key revocation is limited to the number of devices ($O(n)$), the occurrence of a revocation necessitates

 doi:10.20944/preprints202407.2360.v1

the re-generation of keys for all other devices issued with the same key ($O(n^2)$). If the number of devices in the network is large, it may be necessary to further accelerate computations, for example, by parallelizing techniques using GPUs.

Table 4. Combined times of Kafka processing and ABE re-encryption.
(a) - Intel i5-13600KF; (b) - Raspberry Pi 4.

Number of attributes	Kafka and HL processing - Avg time	ABE Encrypt - Avg time	Kafka and ABE Encrypt - Avg time	ABE Decrypt - Avg time		Kafka and ABE Encrypt and Decrypt - Avg time	
		(a)		(a)	(b)	(a)	(b)
1	48.6ms	35.70ms	84.30ms	15.35ms	281.58ms	99.65ms	365.88ms
2	48.6ms	43.80ms	92.4ms	18.12ms	291.37ms	110.52ms	383.77ms
3	48.6ms	46.27ms	94.87ms	15.57ms	301.52ms	110.44ms	396.39ms
5	48.6ms	72.91ms	121.51ms	20.20ms	321.44ms	141.71ms	442.95ms
8	48.6ms	83.20ms	131.8ms	21.29ms	351.73ms	153.09ms	483.53ms

Table 5. Combined times of Kafka processing and ABE re-encryption.
(a) - Intel i5-13600KF; (b) - Raspberry Pi 5.

Number of attributes	Kafka and HL processing - Avg time	ABE Encrypt - Avg time	Kafka and ABE Encrypt - Avg time	ABE Decrypt - Avg time		Kafka and ABE Encrypt and Decrypt - Avg time	
		(a)		(a)	(b)	(a)	(b)
1	48.6ms	35.70ms	84.30ms	15.35ms	201.89ms	99.65ms	286.19ms
2	48.6ms	43.80ms	92.4ms	18.12ms	213.84ms	110.52ms	306.24ms
3	48.6ms	46.27ms	94.87ms	15.57ms	226.58ms	110.44ms	321.45ms
5	48.6ms	72.91ms	121.51ms	20.20ms	250.48ms	141.71ms	371.99ms
8	48.6ms	83.20ms	131.8ms	21.29ms	287.15ms	153.09ms	418.95ms

In Table 6, the average times for new key generation as a function of the number of attributes are presented. The data indicate that delegating these computations to devices with constrained computational capabilities is impractical for large networks. Within the proposed system, while key revocation is limited to the number of devices ($O(n)$), the occurrence of a revocation necessitates the re-generation of keys for all other devices issued with the same key ($O(n^2)$). If the number of devices in the network is large, it may be necessary to further accelerate computations, for example, by parallelizing techniques using GPUs.

Table 6. Time of generating new private key depending on the number of attributes.

Number of attributes	Intel i5-13600KF	Raspberry Pi 5
5	0.422s	1.914s
10	0.795s	2.895s
15	1.371s	3.904s
20	1.928s	4.884s

7. Conclusions

In this study, we introduced a comprehensive data exchange system that includes all of our desired functionalities. Our approach combines ABE with a blockchain-enabled device management system. This combination not only alleviates challenges associated with ABE, such as computational overhead, but also retains its inherent strengths, including distribution properties and the ease of adopting a Data-Centric Security approach, compared to traditional PKI systems.

Our next steps will involve implementing and deploying this system in a real-world setting. While we have estimated delay times resulting from ABE calculations, deploying the system in a real environment will provide more accurate insights, such as network latency considerations.

Additionally, we are intrigued by the prospect of evaluating ABE calculation times on different less powerful devices commonly used in IoT and using lightweight libraries and implementations.

 doi:10.20944/preprints202407.2360.v1

This exploration could offer valuable insights into the system's scalability and performance on resource-constrained platforms.

It should be emphasized that the proposed solution establishes a security layer within information processing systems (e.g., in the Data-Centric Security approach), where security should be evaluated as a complex whole. Further work in this direction is planned for subsequent steps.

Author Contributions: Conceptualization, Krzysztof Kanciak and Zbigniew Zieliński; Data curation, Łukasz Pióro and Jakub Sychowiec; Methodology, Krzysztof Kanciak and Zbigniew Zieliński; Resources, Łukasz Pióro and Jakub Sychowiec; Software, Łukasz Pióro and Jakub Sychowiec; Supervision, Krzysztof Kanciak and Zbigniew Zieliński; Visualization, Łukasz Pióro and Zbigniew Zieliński; Writing – original draft, Łukasz Pióro; Writing – review & editing, Łukasz Pióro, Jakub Sychowiec, Krzysztof Kanciak and Zbigniew Zieliński.

Data Availability Statement: Data is contained within the article.

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

References

1. Abdelzaher, T.; Ayanian, N.; Başar, T.; Diggavi, S.; Diesner, J.; Ganesan, D.; Govindan, R.; Jha, S.; Lepoint, T.; Marlin, B.; Nahrstedt, K.; Nicol, D.; Rajkumar, R.; Russell, S.; Seshia, S.; Sha, F.; Shenoy, P.; Srivastava, M.; Sukhatme, G.; Veeravalli, V. Will Distributed Computing Revolutionize Peace? The Emergence of Battlefield IoT. 2018, pp. 1129–1138. doi:10.1109/ICDCS.2018.00112.
2. Kanciak, K.; Jarosz, M.; Glebocki, P.; Wrona, K. Enabling civil-military information sharing in federated smart environments. 2021 IEEE 7th World Forum on Internet of Things (WF-IoT), 2021, pp. 897–902. doi:10.1109/WF-IoT51360.2021.9595715.
3. Pradhan, M.; Suri, N.; Zielinski, Z.; Tortonesi, M.; Fuchs, C.; Wrona, K.; Furtak, J.; Vasilache, D.; Street, M.; Pellegrini, V.; Benincasa, G.; Morelli, A.; Stefanelli, C.; Casini, E.; Dyk, M. Exploiting smart city IoT for disaster recovery operations. 2018. doi:10.1109/WF-IoT.2018.8355117.
4. Kanciak, K.; Wrona, K. Towards an Auditable Cryptographic Access Control to High-value Sensitive Data. *International Journal of Electronics and Telecommunications* **2020**, *66*, 449–458.
5. Sahai, A.; Waters, B. Fuzzy Identity-Based Encryption. Advances in Cryptology – EUROCRYPT 2005; Cramer, R., Ed.; Springer Berlin Heidelberg: Berlin, Heidelberg, 2005; pp. 457–473.
6. Johnsen, F.; Hauge, M. Interoperable, adaptable, information exchange in NATO coalition operations. *Journal of Military Studies* **2022**, *11*. doi:10.2478/jms-2022-0005.
7. Jansen, N.; Manso, M.; Toth, A.; Chan, K.; Bloebaum, T.; Johnsen, F. NATO Core Services profiling for Hybrid Tactical Networks — Results and Recommendations. 2021, pp. 1–8. doi:10.1109/ICMCIS52405.2021.9486415.
8. Suri, N.; Fronteddu, R.; Cramer, E.; Breedy, M.; Marcus, K.; Velt, R.; Nilsson, J.; Mantovani, M.; Campioni, L.; Poltronieri, F.; Benincasa, G.; Ordway, B.; Peuhkuri, M.; Rautenberg, M. Experimental Evaluation of Group Communications Protocols for Tactical Data Dissemination. 2018, pp. 133–139.
9. De Rango, F.; Potrino, G.; Tropea, M.; Fazio, P. Energy-aware dynamic Internet of Things security system based on Elliptic Curve Cryptography and Message Queue Telemetry Transport protocol for mitigating Replay attacks. *Pervasive and Mobile Computing* **2019**, *61*, 101105. doi:10.1016/j.pmcj.2019.101105.
10. Yang, M.; Margheri, A.; Hu, R.; Sassone, V. Differentially Private Data Sharing in a Cloud Federation with Blockchain. *IEEE Cloud Computing* **2017**, *5*. doi:10.1109/MCC.2018.064181122.
11. Wang, X.; Zha, X.; Ni, W.; Liu, R.P.; Guo, Y.J.; Niu, X.; Zheng, K. Survey on blockchain for Internet of Things. *Computer Communications* **2019**, *136*, 10–29. doi:https://doi.org/10.1016/j.comcom.2019.01.006.
12. Guo, S.; Wang, F.; Zhang, N.; Qi, F.; Xuesong, Q. Master-slave chain based trusted cross-domain authentication mechanism in IoT. *Journal of Network and Computer Applications* **2020**, *172*, 102812.
13. Xu, L.; Chen, L.; Gao, Z.; Fan, X.; Suh, T.; Shi, W. DIoTA: Decentralized-Ledger-Based Framework for Data Authenticity Protection in IoT Systems. *IEEE Network* **2020**, *34*, 38–46. doi:10.1109/MNET.001.1900136.
14. Khalid, U.; Asim, M.; Baker, T.; Hung, P.; Tariq, M.A.; Rafferty, L. A decentralized lightweight blockchain-based authentication mechanism for IoT systems. *Cluster Computing* **2020**, *23*. doi:10.1007/s10586-020-03058-6.
15. Müller, S.; Katzenbeisser, S.; Eckert, C. Distributed Attribute-Based Encryption. Information Security and Cryptology – ICISC 2008; Lee, P.J.; Cheon, J.H., Eds.; Springer Berlin Heidelberg: Berlin, Heidelberg, 2009; pp. 20–36.

16. Jiang, J.; Gao, Y.; Gong, Y.; Jiang, Z. A Blockchain Copyright Protection Scheme Based on CP-ABE Scheme with Policy Update. *Sensors* **2024**, *24*. doi:10.3390/s24144493.
17. Lu, Y.; Feng, T.; Liu, C.; Zhang, W. A Blockchain and CP-ABE Based Access Control Scheme with Fine-Grained Revocation of Attributes in Cloud Health. *Computers, Materials & Continua* **2024**, *78*, 2787–2811. doi:10.32604/cmc.2023.046106.
18. Gondalia, A.; Dixit, D.; Parashar, S.; Raghava, V.; Sengupta, A.; Sarobin, V. IoT-based Healthcare Monitoring System for War Soldiers using Machine Learning. *Procedia Computer Science* **2018**, *133*, 1005–1013. doi:10.1016/j.procs.2018.07.075.
19. V, S.; R, S.; B, A.; V, V.S.; Vigneswari, P. IoT based Healthcare Monitoring and Tracking System for Soldiers using ESP32. 2022 6th International Conference on Computing Methodologies and Communication (ICCMC), 2022, pp. 377–381. doi:10.1109/ICCMC53470.2022.9754076.
20. Wrona, K. Securing the Internet of Things a military perspective. 2015 IEEE 2nd World Forum on Internet of Things (WF-IoT), 2015, pp. 502–507. doi:10.1109/WF-IoT.2015.7389105.
21. Sueur, P.L. The Felin soldier system: a tailored solution for networked operations. SPIE Defense + Commercial Sensing, 2007.
22. Dietterle, R. The future combat systems (FCS) overview. 2005, pp. 3269 – 3273 Vol. 5.
23. Systems, P.M.G.S. Nett Warrior Interconnect Architecture White Paper. 2017.
24. Kanciak, K.; Wrona, K.; Jarosz, M. Secure Onboarding and Key Management in Federated IoT Environments. 2022, pp. 627–634. doi:10.15439/2022F173.
25. Sychowiec, J.; Zielinski, Z. An Experimental Framework for Secure and Reliable Data Streams Distribution in Federated IoT Environments. 2023, pp. 769–780. doi:10.15439/2023F3882.
26. Belguith, S.; Kaaniche, N.; Hammoudeh, M. Analysis of attribute based cryptographic techniques and their application to protect cloud services. *Transactions on Emerging Telecommunications Technologies* **2022**, *33*. doi:10.1002/ett.3667.
27. Tomida, J.; Kawahara, Y.; Nishimaki, R. Fast, Compact, and Expressive Attribute-Based Encryption. Cryptology ePrint Archive, Paper 2019/966, 2019. https://eprint.iacr.org/2019/966.
28. Susan Symington, W. Polk, M.S. Trusted Internet of Things (IoT) Device Network-Layer Onboarding and Lifecycle Management, 2020.
29. Article code repository. https://github.com/mojitax/Application-of-Attribute-Based-Encryption-in-Military-Internet-of-Things-Environment. Accessed: 2024-07-19.
30. CIRCL github repository. https://github.com/cloudflare/circl. Accessed: 2024-04-14.
31. Barreto, P.; Lynn, B.; Scott, M. Constructing Elliptic Curves with Prescribed Embedding Degrees. 2002, Vol. 2576. doi:10.1007/3-540-36413-7_19.
32. BLS12-381 Curve Description. https://electriccoin.co/blog/new-snark-curve/. Accessed: 2024-04-17.
33. Guillevic, A.; Masson, S.; Thomé, E. Cocks–Pinch curves of embedding degrees five to eight and optimal ate pairing computation. *Designs, Codes and Cryptography* **2020**, *88*. doi:10.1007/s10623-020-00727-w.
34. Boulogeorgos, A.A.A.; Bouzouita, M.; Ksentini, A.; Fossorier, M. Analysis of Web-Based IoT through Heterogeneous Networks. *Sensors* **2022**, *22*, 509. doi:10.3390/s22020509.