

Article

Comparative Analysis of Attribute-Based Encryption Schemes for Special Internet of Things Applications

Łukasz Pióro , Krzysztof Kanciak  and Zbigniew Zieliński 

Faculty of Cybernetics, Military University of Technology, 00-908 Warsaw, Poland;
krzysztof.kanciak@wat.edu.pl (K.K.); zbigniew.zielinski@wat.edu.pl (Z.Z.)

* Correspondence: lukasz.pioro@wat.edu.pl

Abstract

Attribute-based encryption (ABE) is an advanced public key encryption mechanism that enables the precise control of access to encrypted data based on attributes assigned to users and data. Attribute-based access control (ABAC), which is built on ABE, is crucial in providing dynamic, fine-grained, and context-aware security management in modern Internet of Things (IoT) applications. ABAC controls access based on attributes associated with users, devices, resources, and environmental conditions rather than fixed roles, making it highly adaptable to the complex and heterogeneous nature of IoT ecosystems. ABE can significantly improve the security and manageability of modern military IoT systems. Nevertheless, its practical implementation requires obtaining a range of performance data and assessing the additional overhead, particularly regarding data transmission efficiency. This paper provides a comparative analysis of the performance of two cryptographic schemes for attribute-based encryption in the context of special Internet of Things (IoT) applications. This applies to special environments, both military and civilian, where infrastructure is unreliable and dynamic and decisions must be made locally and in near-real time. From a security perspective, there is a need for strong authentication, precise access control, and a zero-trust approach at the network edge as well. The CIRCL scheme, based on traditional pairing-based ABE (CP-ABE), is compared with the newer Covercrypt scheme, a hybrid key encapsulation mechanism with access control (KEMAC) that provides quantum resistance. The main goal is to determine which scheme scales better and meets the performance requirements for two different scenarios: large corporate networks (where scalability is key) and tactical edge networks (where minimal bandwidth and post-quantum security are paramount). The benchmark results are used to compare the operating costs in detail, such as the key generation time, message encryption and decryption times, public key size, and cipher overhead, showing that Covercrypt provides a reduction in ciphertext overhead in tactical scenarios, while CIRCL offers faster decryption throughput in large-scale enterprise environments. It is concluded that the optimal choice depends on the specific constraints of the operating environment.



Academic Editor: Lei Shu

Received: 26 November 2025

Revised: 18 January 2026

Accepted: 4 February 2026

Published: 5 February 2026

Copyright: © 2026 by the authors.

Licensee MDPI, Basel, Switzerland.

This article is an open access article distributed under the terms and conditions of the [Creative Commons Attribution \(CC BY\)](https://creativecommons.org/licenses/by/4.0/) license.

Keywords: Internet of Things; security; attribute-based encryption; attribute-based access control

1. Introduction and Related Works

Currently, we can observe many so-called special applications of the Internet of Things (IoT) in both the civilian and military domains, where IoT is used in environments characterized by unreliable communication infrastructure, high event dynamics, and the need

to make decisions in near-real time. In practice, this means that data must be processed locally, close to IoT end devices and users, with minimal dependence on a central data center. Examples of special applications in civilian areas include rescue and crisis management, public safety, and smart cities [1]. Rescue teams often have to build ad hoc mobile processing nodes in vehicles (tactical clouds) to analyze data from drones, cameras, and sensors without permanent Internet access. The use of IoT in military operations improves effectiveness by enabling the integration, analysis, and exchange of data between sensors, command and control systems, combat platforms, and specialized field devices. The integration of IoT into military environments improves situational awareness [2], enables faster commander decision making, improves resource management efficiency, and provides real-time decision support, all of which are crucial in the context of today's asymmetric and multi-dimensional threats [3].

However, the widespread implementation of IoT systems for special applications poses significant security challenges, including vulnerability to cyberattacks, issues with the integrity and confidentiality of transmitted data, and difficulties in managing access in decentralized operating environments [4,5]. In response to these problems, research is being conducted on advanced protection mechanisms, among which attribute-based encryption (ABE) and attribute-based access control (ABAC) play an important role [6–11].

ABE is an advanced cryptographic scheme that enables fine-grained access control to encrypted data based on attributes assigned to users, resources, or the environment [12]. In the ABE model, both users' private keys and ciphertexts (encrypted messages) are associated with attributes that describe entities, roles, or access contexts. Decryption is possible only if the user's attributes satisfy the access policy defined when the message was encrypted. There are two basic types of ABE [12]:

- In key-policy attribute-based encryption (KP-ABE), the access policy is embedded in the user's private key, and the encrypted data are marked with a set of attributes. The user will only decrypt the data if the ciphertext's attribute set matches the user's key policy.
- In ciphertext-policy attribute-based encryption (CP-ABE), the opposite is true—the access policy is associated with the ciphertext, and the user's private key contains a set of attributes. Decryption occurs if the user has the required set of attributes consistent with the policy specified during encryption.

ABE provides a solid technical foundation for implementing ABAC models in distributed, dynamic environments, such as military IoT, where flexible, scalable access control is essential [13,14]. Thus, ABE can be seen as a cryptographic implementation of the ABAC concept.

ABAC is recommended for military IoT systems due to its ability to enforce security policies in a highly granular, adaptive, and robust manner, being well suited to the dynamic and sensitive nature of military operations and the hostile environments in which they are conducted [15]. This is expected to bring several benefits, including improved security in the IoT through capabilities such as precise, contextual access control, the mitigation of insider threats, and zero-trust requirements. ABAC enables precise access control by evaluating multiple attributes (e.g., user role, device, location, time, mission context), rather than static roles or simple user lists. This flexibility allows military systems to enforce context-dependent restrictions, tailoring access in real time to operational needs and threat levels. By evaluating actual attributes rather than predefined roles, ABAC reduces the risks of insider threats and privilege escalation by ensuring that only authorized attribute-based actions are allowed—even after credential compromises.

Special applications of the IoT often involve federated systems [16] across diverse partners (coalitions, allied units, civilian interfaces) [17], with varying clearance levels and

data sensitivity. ABAC's attribute-centric model supports secure, fine-grained information sharing and coalition operations while respecting policy requirements and minimizing data leakage between partners. The IoT environment in military contexts is highly dynamic, with devices and users frequently joining, leaving, or changing roles and locations.

ABAC can be tightly integrated with cryptographic schemes, especially ABE, to enforce object-level protection and enable a data-centric security [18] approach in battlefield IoT and military environments [19]. With ABE, data (objects) can be encrypted using an access policy based on attributes (such as user roles, clearance levels, device types), directly mapping to ABAC policies. Only users or devices with matching attributes (encoded in their cryptographic keys) can decrypt and access the data, regardless of the location of the network or intermediate storage. This ensures secure end-to-end protection at the object level, not just the perimeter. Data are encrypted as soon as they are generated or ingested by IoT devices, with ABAC rules embedded in the encryption policy (ciphertext-policy ABE, or CP-ABE). Decryption is possible only if the requesting user's or system's attributes satisfy the policy embedded in the encrypted object, eliminating the reliance on a single enforcement point and supporting multi-domain coalition environments.

Although the potential applications of ABAC in special IoT systems are emerging, the practical deployment of this method requires further research and experimentation to address emerging questions:

- Which of the known schemes will be most suitable for use in special IoT systems and in which specific use cases?
- How does performance depend on a range of typical parameters for special IoT?
- Which scheme is better to use for protection in the edge IoT area and which in larger-scale networks?
- Can ABAC policies be scaled to large distributed networks, effectively managing complex relationships between numerous entities without the unmanageable overhead found in traditional models such as role-based access control (RBAC)?
- How well is ABE and ABAC integration suited for the special IoT due to support for dynamic, large-scale, and context-driven environments with fine-grained control and reduced management overhead?

To answer the above research questions, a comparative performance analysis between the CIRCL scheme based on CP-ABE pairing and the hybrid post-quantum KEMAC [20] Covercrypt has been conducted. This research aimed to obtain key information on the implementation of data-centric security (DCS) across the various operating environments of the civilian and military Internet of Things.

The CIRCL library implements an efficient CP-ABE scheme, referred to as tkn20, based on the work by Tomida et al. [21]. This scheme guarantees IND-CCA2 security and supports expressive policies. In [7], it has been proven that, under certain assumptions, it is also applicable for resource-constrained IoT devices, which is particularly relevant in demanding military use cases.

In contrast, Covercrypt provides a hybrid post-quantum secure encryption scheme that combines key encapsulation with attribute-based policies. This architecture enables fast encryption and robust access control, which are essential for special IoT systems requiring quantum-resilient security [22]. Benchmarking studies, such as those conducted by de la Piedra et al. [23], suggest that pairing-based CP-ABE schemes like tkn20 often incur higher computational costs than hybrid schemes like Covercrypt, which demonstrate advantages in edge computing scenarios.

The integration of ABAC and ABE allows for the scalable, flexible, and secure management of dynamic multi-domain IoT networks. This capability helps to mitigate insider threats and is fundamental to implementing zero-trust environments [24]. Furthermore,

this integration aligns with the data-centric security approach, ensuring data protection from their origin throughout their lifecycle and enabling granular access control [25]. The adoption of DCS is currently under consideration by multiple entities, including the military sector [26]. This research also aligns with the recent NIST standardization of device onboarding in IoT networks [27]. However, further research is required to benchmark these schemes under special IoT application-specific constraints and to optimize their deployment strategies.

This paper contributes to the field by providing a comparative performance analysis between the CIRCL CP-ABE scheme and Covercrypt. This analysis advances our understanding of suitable data-centric security approaches specifically tailored to specialized IoT environments. This study represents the first direct comparison of these two schemes and their potential applications, highlighting their respective strengths and weaknesses. Although benchmarks for various ABE libraries, such as those in [28], have been provided previously, these two specific libraries have not yet been evaluated together. Furthermore, this study defines operational scenarios based on real-world applications in the civilian and military domains, providing the first direct evaluation of these cryptographic schemes with regard to specific IoT system requirements.

The remainder of the paper is organized as follows. Section 2 briefly characterizes the foundations of modern data access control, particularly attribute-based access control (ABAC) and the data-centric security paradigm, as well as ETSI's fundamentals for the standardization of post-quantum attribute-based encryption, and briefly explains the foundations and basic phases involved in applying ciphertext-policy attribute-based encryption (CP-ABE) and Covercrypt as an attribute-based key encapsulation mechanism (KEM). Section 3 presents the methodology and results of the comparison tests. Section 3.9 provides an analysis of the complexity and scalability of the compared schemes in civilian and military IoT, focusing on key parameters. Both ciphertext-policy attribute-based encryption (CP-ABE, as implemented by CIRCL) and the key encapsulation mechanism with access control (KEMAC, as implemented by Covercrypt) are evaluated based on their operational costs as a function of the system parameters. In Section 3.10, the trade-offs between the highly scalable pairing-based CP-ABE (CIRCL) and the bandwidth-efficient, quantum-safe KEMAC (Covercrypt) are presented in the two distinct IoT operational scenarios. In Section 4, the results are summarized and directions for further work are identified.

2. Background

2.1. The Foundations of Modern Data Access Control

2.1.1. Attribute-Based Access Control

Attribute-based access control (ABAC) is defined by the National Institute of Standards and Technology (NIST) as an access control model where authorizations to perform operations on resources are granted to subjects based on the evaluation of policies against the attributes of subjects, resources, and the environment [29].

The core of an ABAC model is a set of policies that combine and evaluate these attributes using logical rules. The fundamental components are as follows:

- Subject attributes: Characteristics of the user (e.g., identity, clearance, department).
- Resource attributes: Characteristics of the object being accessed (e.g., creation date, data classification, owner).
- Action attributes: Characteristics of the operation being attempted (e.g., 'read', 'write', 'delete').
- Environmental attributes: Contextual factors such as time of day, location of access, or device security posture [29].

This structure allows for the creation of highly expressive access control policies (e.g., “Allow doctors to view medical records they created within their own hospital during work hours”), which are essential for securing complex, distributed systems [29]. In the context of this report, the focus is on cryptographically enforced ABAC. Here, the policy decision point (PDP) and policy enforcement point (PEP) are effectively merged into the cryptographic algorithms themselves. Two approaches can be distinguished:

KP-ABE Key-Policy ABE: Attributes are associated with the ciphertext, and policies are embedded in the user’s key. This is useful for database queries or content-centric routing.

CP-ABE Ciphertext-Policy: Attributes are associated with the user’s key, and policies are embedded in the ciphertext. This is the most natural fit for DCS, as it allows the data owner (the encryptor) to decide who can access the data at the moment of creation.

Both CIRCL (implementing tkn20) and Covercrypt utilize the CP-ABE model (or its functional equivalent in KEMAC), placing control in the hands of the data originator.

2.1.2. The Need for Data-Centric Security in Coalition Warfare

Modern warfare has evolved into network-centric warfare (NCW), where information superiority is as decisive as firepower. The integration of IoT devices—ranging from unattended ground sensors (UGS) to soldier-worn biometric monitors—generates a torrent of sensitive data that must be shared across disparate branches and coalition partners. Traditional “castle-and-moat” security architectures, which rely on securing the network perimeter (e.g., via VPNs or firewalls), are fundamentally incompatible with the dynamic, ad hoc nature of coalition operations. In these environments, the network infrastructure itself may be untrusted or shared with partners possessing varying clearance levels.

Data-centric security (DCS) addresses this by decoupling data protection from the transport layer. In a DCS model, the data are encrypted at the source (the sensor or soldier) and remain encrypted until they reach an authorized recipient, regardless of the intermediary nodes that they traverse. The access control policy is not enforced by a central server (which represents a single point of failure and a latency bottleneck) but is mathematically embedded within the encrypted data themselves.

Attribute-based encryption (ABE) is the cryptographic cornerstone of DCS. Unlike traditional Public Key Infrastructure (PKI), which encrypts data for a specific identity (e.g., “encrypt for Serial Number 12345”), ABE encrypts data for a policy defined over a set of attributes (e.g., “encrypt for (Role: Medic AND Clearance: Secret) OR (Role: Commander)”). This allows for “one-to-many” encryption and enables the system to grant access to users who may not even exist at the time of encryption, provided that they eventually possess the requisite credentials. This capability is essential for the dynamic join-and-leave behavior of tactical units in the field. The data-centric security (DCS) paradigm marks a strategic shift from protecting network perimeters to protecting the data themselves. In this model, security is not an external shell but an intrinsic property of the data object. The objective is to ensure that the data remain protected regardless of their location, whether they are at rest in a database, in transit over a network, or in use within an application.

DCS is realized by binding security policies, primarily through cryptography, directly to the data. Key mechanisms enabling this paradigm include strong encryption, embedded access control logic, and data classification. By making data self-defending, DCS addresses the shortcomings of perimeter-based models, which become ineffective once a breach occurs or when data are shared with external partners. Attribute-based encryption (ABE) is considered a key cryptographic enabler for achieving a true data-centric architecture, as it allows access policies to be inseparable from the encrypted data [12].

2.2. ETSI's Standardization of Post-Quantum, Attribute-Based Encryption

2.2.1. The Quantum Threat and the Need for a New Standard

The emergence of practical quantum computing poses a catastrophic threat to widely used public key cryptographic algorithms, such as RSA and ECC [30]. A sufficiently powerful quantum computer, leveraging Shor's algorithm, could break the mathematical foundations of this cryptography, rendering vast amounts of secured data and communications vulnerable. This threat is particularly acute for data with long-term confidentiality requirements due to "harvest now, decrypt later" attacks, where an adversary archives encrypted data today with the intent of decrypting them in the future [31].

Recognizing this, standardization bodies like ETSI initiated efforts to define cryptographic systems that are secure against both classical and quantum computers. The goal was to establish a clear standard for "crypto-agility"—the ability to transition to quantum-resistant algorithms—while simultaneously enhancing the functionality of cryptographic primitives to better serve modern access control needs [32]. The strategic utility of ABE is currently threatened by the rapid advancement of quantum computing. Standard ABE schemes rely on the hardness of the Decisional Bilinear Diffie–Hellman (DBDH) problem over elliptic curves. Shor's algorithm proves that a sufficiently powerful quantum computer could solve the discrete logarithm problem in polynomial time, effectively breaking all pairing-based cryptography. While cryptographically relevant quantum computers (CRQCs) may not be available for another decade, the threat is immediate due to "harvest-now, decrypt-later" strategies. Adversaries are actively intercepting and archiving encrypted military communications today, with the intent of decrypting them once quantum capabilities mature. Consequently, military systems designed with a lifecycle of 10–20 years must integrate post-quantum cryptography (PQC) immediately to protect long-term intelligence. This necessity has driven the European Telecommunications Standards Institute (ETSI) to standardize Covercrypt, a hybrid scheme that combines classical elliptic curve cryptography with quantum-resistant Lattice-based primitives (specifically, variants of the NIST finalist Kyber) [33]. By layering these protections, Covercrypt ensures that data remain secure as long as at least one of the underlying mathematical problems remains hard.

2.2.2. Covercrypt: A Hybrid KEMAC Solution

In response to these challenges, ETSI published the technical specification ETSI TS 104 015, which standardizes a hybrid, quantum-safe cryptographic scheme known as Covercrypt [32]. The scheme is formally defined as a key encapsulation mechanism with access control (KEMAC).

Its hybrid nature is a critical feature for a secure transition. It combines a classical, well-established algorithm with a post-quantum algorithm from the finalists/alternates of the NIST PQC standardization process. Security is maintained as long as at least one of the component algorithms remains unbroken, thus providing resilience against unforeseen breaks in either primitive [20,34]. Most importantly, Covercrypt natively implements ciphertext-policy attribute-based encryption, as detailed in Section 2.4. This allows a data originator to encrypt a session key under a specific access policy (e.g., (role = 'auditor' AND region = 'EMEA') OR clearance = 'L3'). Only users whose cryptographic keys are associated with attributes that satisfy this policy can decapsulate the session key and decrypt the corresponding data [32].

2.2.3. Connecting Covercrypt with ABAC and DCS

The ETSI standardization of Covercrypt provides a formal, robust cryptographic foundation for implementing ABAC and realizing the DCS paradigm.

- Link to ABAC: Covercrypt is a direct cryptographic enforcement mechanism for ABAC. It translates abstract ABAC policies into immutable cryptographic rules embedded within the ciphertext itself. The authorization check is not performed by a fallible external policy decision point but is an intrinsic part of the decryption process. This eliminates entire classes of vulnerabilities related to policy enforcement and ensures that access control is absolute.
- Link to DCS: Covercrypt is a quintessential enabling technology for data-centric security. By binding an attribute-based access policy to the data through encryption, the data become a self-contained, self-protecting entity. Their security is inherent and persists across different environments—cloud storage, third-party applications, or local devices. This directly addresses the core objective of DCS: to protect data throughout their lifecycle, ensuring that only authorized individuals can access them, regardless of where they reside.

Thus, ETSI TS 104 015 not only provides a forward-looking solution to the quantum threat but also standardizes a powerful primitive for building the next generation of secure, fine-grained, and data-centric information systems.

2.3. Ciphertext-Policy Attribute-Based Encryption

Ciphertext-policy attribute-based encryption (CP-ABE) is an advanced cryptographic primitive that enables fine-grained access control over encrypted data. In a CP-ABE system, a user's private key is associated with a set of descriptive attributes, and a ciphertext is encrypted with an access policy defined over these attributes. A user can only decrypt a ciphertext if the attributes associated with their private key satisfy the policy embedded within the ciphertext.

The tkn20 scheme [35], available in Cloudflare's CIRCL library [36], is a modern and efficient implementation based on the foundational work of Waters. A trusted key authority (KA) is responsible for generating public parameters in the system and a master secret key (MSK). The KA then uses the MSK to issue personalized private keys to users based on their attributes.

2.3.1. Setup

Before any keys can be issued or data encrypted, a trusted entity known as the key authority (KA) must initialize the system. This is accomplished through a 'Setup' algorithm that creates the essential public and private parameters for the entire environment.

The 'Setup' algorithm generates the following:

- Public Parameters (PP): These parameters are made publicly available and are used by anyone who wishes to encrypt data. The PP typically consists of

$$PP = (g, g^\beta, H)$$

where g is a generator of a cyclic group $\mathbb{G}$ of order p , $\beta \in \mathbb{Z}_p$ is a randomly chosen secret exponent, and H is a public hash function that maps attribute strings to elements in $\mathbb{G}$.

- Master Secret Key (MSK): This key is the root of trust for the entire system and must be kept secret by the key authority at all times. The MSK consists of

$$MSK = (\beta, g^\alpha)$$

where $\alpha \in \mathbb{Z}_p$ is another randomly chosen secret exponent. The MSK empowers the KA to issue private keys to users.

2.3.2. User Key Generation

The generation of a user's private key (SK) is a critical process managed exclusively by the key authority. It involves combining the system's master secret key with the user's specific attribute set. The core of the process, particularly its security against collusion, lies in the introduction of a unique, random secret for each user.

Let the system's public parameters be (g, g^β, H) , where g is a generator of a cyclic group $\mathbb{G}_0$, β is a secret exponent, and H is a hash function that maps attributes to elements of $\mathbb{G}_0$. The master secret key (MSK) is (β, g^α) , where α is another secret exponent.

The key generation algorithm, often denoted as 'KeyGen(MSK, A)', where A is the user's attribute set, proceeds as follows:

1. Generate a User-Specific Random Secret: The KA first selects a random secret $t \in \mathbb{Z}_p$ uniformly at random. This value is unique for each key generation instance and serves as the cornerstone of collusion resistance.
2. Compute the Base Key Component: A base component of the private key, K_0 , is computed. This component binds the user's secret t with the master secrets α and β .

$$K_0 = g^\alpha \cdot g^{\beta t}$$

This component is essential for the decryption process and ensures that any valid key has been properly derived from the MSK.

3. Compute Attribute-Specific Components: For each attribute j in the user's attribute set A , a corresponding key component K_j is computed. Crucially, each of these components is also blinded by the same user-specific secret t .

$$K_j = H(j)^t$$

Here, $H(j)$ is the result of hashing the attribute string (e.g., "role: doctor") to a group element.

The final user private key SK is the collection of the base component and all attribute-specific components:

$$SK = (K_0, \{K_j\}_{j \in A})$$

This structure ensures that all parts of a user's key are mathematically linked through the single secret value t .

2.3.3. Encryption

An entity wishing to encrypt a message M does so by using the public parameters and defining an access policy $\mathbb{A}$. The access policy is a logical formula (e.g., ("role:doctor" AND "department:cardiology") OR "role:admin") that specifies which attributes are required for decryption. This policy is typically represented internally as a linear secret sharing scheme (LSSS).

The Encrypt(PP, M, $\mathbb{A}$) algorithm proceeds as follows:

1. Select an Encryption Secret: The encryptor chooses a random, single-use secret $s \in \mathbb{Z}_p$. This secret will be used to encrypt the message and will be embedded in the ciphertext.
2. Encrypt the Message: The main part of the ciphertext, C' , is created by binding the message M with the master secret component α and the encryption secret s . This is achieved using a pairing function e :

$$C' = M \cdot e(g, g)^{\alpha s}$$

The term $e(g, g)^{as}$ acts as a one-time symmetric key (a blinding factor) that masks the message.

3. Embed the Secret: A second component, C_0 , is created to carry the encryption secret s , blinded by the master secret β :

$$C_0 = g^{\beta s}$$

4. Encode the Policy: For each attribute i that is part of the access policy $\mathbb{A}$, a corresponding ciphertext component C_i is generated. These components securely embed the shares of the secret s according to the LSSS representation of the policy. For each attribute i , this resembles

$$C_i = H(i)^s$$

(This is a simplification; in an LSSS, each C_i corresponds to a row in the policy matrix and involves more complex calculations, but the core idea is binding the attribute to the secret s).

The final ciphertext CT consists of the access policy $\mathbb{A}$ and all generated components:

$$CT = (\mathbb{A}, C', C_0, \{C_i\})$$

2.3.4. Decryption

A user with private key SK can attempt to decrypt a ciphertext CT . The process is successful only if the user's attributes A satisfy the ciphertext's access policy $\mathbb{A}$.

The 'Decrypt(SK, CT)' algorithm involves the following steps:

1. Policy Check: First, the algorithm determines if the attributes A in the user's key SK satisfy the policy $\mathbb{A}$ in the ciphertext CT . If not, decryption fails immediately.
2. Reconstruct the Blinding Factor: If the policy is satisfied, the user can use a specific subset of their attribute key components $\{K_j\}$ that corresponds to the attributes needed to fulfil the policy. Through a series of pairing operations involving their key components $(K_0, \{K_j\})$ and the ciphertext components $(C_0, \{C_i\})$, the user reconstructs the original blinding factor $e(g, g)^{as}$.

The core mathematical step involves combining the components in such a way that cancels out the user's secret t and the policy's secret shares, leaving only the required term. A simplified view of the final pairing result is

$$\frac{e(K_0, C_0)}{\text{Pairings}(\{K_j\}, \{C_i\})} = \frac{e(g^\alpha \cdot g^{\beta t}, g^{\beta s})}{\dots} = e(g, g)^{as}$$

The denominator, representing the pairings of attribute components, is calculated using constants derived from the LSSS matrix to correctly combine the shares of s and cancel out the $e(g, g)^{\beta ts}$ term.

3. Recover the Message: Once the blinding factor $B = e(g, g)^{as}$ has been successfully reconstructed, recovering the original message involves simple division:

$$M = \frac{C'}{B} = \frac{M \cdot e(g, g)^{as}}{e(g, g)^{as}}$$

If a user's attributes do not satisfy the policy, they will not have the necessary set of K_j components to correctly run the reconstruction algorithm. The mathematical operations will not produce the correct blinding factor, and the result will be garbage data instead of the original message M .

2.3.5. The Collusion Resistance Mechanism

A collusion attack occurs when multiple users pool their private keys to decrypt a ciphertext that none of them could decrypt individually. For example, User 1 with attribute {"doctor"} and User 2 with attribute {"cardiology"} might try to combine their keys to satisfy an access policy of "doctor" AND "cardiology". The key structure of the tkn20 scheme makes this computationally infeasible.

The security comes from the fact that each user's key is randomized with a different secret. Let us analyze the keys of two distinct users.

- User 1 (attributes $A_1 = \{\text{"doctor"}\}$) is assigned a random secret t_1 . Their private key SK_1 is

$$SK_1 = (K_{0,1} = g^\alpha \cdot g^{\beta t_1}, \quad K_{\text{doctor},1} = H(\text{"doctor"})^{t_1})$$

- User 2 (attributes $A_2 = \{\text{"cardiology"}\}$) is assigned a random secret t_2 . Their private key SK_2 is

$$SK_2 = (K_{0,2} = g^\alpha \cdot g^{\beta t_2}, \quad K_{\text{cardiology},2} = H(\text{"cardiology"})^{t_2})$$

Since the KA chooses secrets randomly, it is overwhelmingly likely that $t_1 \neq t_2$.

During decryption, pairing-based cryptography is used to check if the attributes satisfy the policy. This process involves pairing operations that must cancel out certain secret terms in the exponent to reconstruct a blinding factor. For this to succeed, all attribute key components used in the decryption equation must be derived from the same exponent.

If the colluding users attempt to mix their key components, they would be using $H(\text{"doctor"})^{t_1}$ from User 1 and $H(\text{"cardiology"})^{t_2}$ from User 2. The decryption algorithm, which expects all attribute components to be raised to the power of a single t , will fail. The mathematical checks will not reach validation because the underlying exponents (t_1 and t_2) are different. There is no known algebraic operation that allows attackers to transform $H(j_1)^{t_1}$ and $H(j_2)^{t_2}$ into a valid set $\{H(j_1)^{t'}, H(j_2)^{t'}\}$ for some consistent t' without knowing t_1 and t_2 , which are secret.

2.4. Covercrypt Overview

Covercrypt is an attribute-based key encapsulation mechanism (KEM) designed to enable fine-grained access control, privacy of access policies, hybrid (classical + post-quantum) security, and traceability [20]. It supports Boolean combinations—"AND" and "OR"—over multi-dimensional attributes (e.g., "Department::Sales AND Country::FR"). Ciphertexts are tagged with a set of target attributes, and users hold decryption keys tied to certain attribute sets.

Covercrypt operates through four intertwined phases, namely master setup, user key generation, encryption, and decryption, elegantly combining classical and post-quantum KEMs under an attribute-based paradigm. In the setup phase, the system defines a complete universe of "rights"; each right corresponds to a specific combination of attributes drawn from predefined dimensions, such as department, country, or rank. For each possible right, the authority generates an independent key pair using a KEM (e.g., X25519 or Kyber). These sub-key pairs collectively form a master public key, while the secret halves form the master secret key.

The encryption phase begins with the encryptor defining an access policy and similarly converting it into an explicit right set. Then, for each right in this set, the corresponding public key is used to perform KEM encapsulation, yielding a ciphertext component E_a and a shared secret K_a . A single random seed S is drawn and masked against each K_a to generate $F_a = S \oplus K_a$. All pairs (E_a, F_a) , together with a consistency tag, form the encryption header.

Finally, a symmetric key is derived from S , which is used to encrypt the actual message payload following the KEM-DEM paradigm.

In decryption, the user iterates through their secret sub-keys, testing each one against the corresponding ciphertext component. For each right candidate a , they attempt to restore K_a via decapsulation, recover S using F_a , and verify the consistency tag. Once a matching right is found, the correct symmetric key is derived and used to decrypt the payload. This “early abort” process preserves policy privacy and efficiency.

To support post-quantum security, Covercrypt constructs each right’s KEM in a hybrid fashion: combining a classical scheme with a PQ scheme ensures security as long as at least one underlying primitive remains secure. Since the scheme is modular, any new KEM can be plugged in without altering the overall structure. Additionally, Covercrypt embeds a traceable ElGamal KEM (TKEM) into its hybrid AB-KEMAC. Each user receives a unique secret share enabling decrypt, and any session key successfully recovered can be cryptographically tied back to the specific user key used—without revealing any identity during standard operation.

During user key generation, the key-generating entity takes a user policy, expressed as a Boolean formula over attributes, and converts it into a disjunctive set of rights that satisfy the policy. Every right in this expansion corresponds to one of the subkeys in the master secret key, which are then securely packaged and issued to the user.

The algorithm is explained in detail in the following subsections, in accordance with [34].

2.4.1. Setup

Let Ω be the set of rights and t a threshold for traceability. The security parameter is denoted by κ . The setup is as follows:

1. Sample $sk_k \xleftarrow{\$} \mathbb{Z}_p$ for $k \in \{1, \dots, t\}$, and set $P_k \leftarrow sk_k \cdot P$.
2. Sample $\alpha_1, \dots, \alpha_t \xleftarrow{\$} \mathbb{Z}_p$ and define

$$s = \sum_{k=1}^t \alpha_k \cdot sk_k \quad \text{and} \quad H = \sum_{k=1}^t \alpha_k P_k$$

with the constraint that $s \neq 0$.

3. Initialize the set of user identities:

$$\text{ID} \leftarrow \emptyset$$

4. Define the tracing secret key:

$$\text{tsk} = (s, (sk_k)_{k=1}^t, \text{ID})$$

5. Define the tracing public key:

$$\text{tpk} = (H, (P_k)_{k=1}^t)$$

6. Initialize the set of users’ secret keys that show their permissions:

$$\text{UP} \leftarrow \emptyset$$

7. For each right S_i with index i in Ω ,

- 7.1. Sample $(pk_i, sk_i) \leftarrow \text{KEM.KeyGen}(1^\kappa)$, and $x_i \xleftarrow{\$} \mathbb{Z}_p$

7.2. Compute

$$X_i \leftarrow x_i \cdot P, \quad H_i \leftarrow s \cdot X_i$$

7.3. Set

$$pk'_i = (H_i, pk_i), \quad sk'_i = (x_i, sk_i)$$

The global public key is

$$\text{MPK} = (\text{tpk}, \{pk'_i\}_i)$$

and the master secret key is

$$\text{MSK} = (\text{tsk}, \{sk'_i\}_i, \text{UP})$$

2.4.2. User Key Generation

Given a username U , a set Y of rights assigned to U , and the master secret key MSK , the key generation procedure $\text{KeyGen}(\text{MSK}, U, Y)$ proceeds as follows:

1. Parse the master secret key:

$$\text{MSK} = (\text{tsk}, \{sk'_i\}_i, \text{UP})$$

2. Sample a random tuple $(\beta_k)_k$ such that

$$s = \sum_k \beta_k \cdot sk_k$$

and set the user's secret identifier:

$$\text{uid} \leftarrow (\beta_k)_k$$

3. Define the user's secret key:

$$\text{usk} \leftarrow (\text{uid}, \{sk'_j\}_{j \in Y})$$

4. Update the tracing secret key tsk by appending (U, uid) to the identity set ID .
5. Update the master secret key MSK by appending usk to the permission set UP .

2.4.3. Encryption

Given a set X of rights and the public key MPK , the encryption algorithm $\text{Enc}(\text{MPK}, X)$ proceeds as follows:

1. Parse the public key:

$$\text{MPK} = (\text{tpk}, \{pk'_i\}_i)$$

2. Let $\mathcal{K}$ denote the key space of KEM, and sample

$$S \xleftarrow{\$} \mathcal{K}$$

3. Set

$$r \leftarrow \mathcal{G}(S), \quad c_k \leftarrow r \cdot P_k \quad \text{for each } k, \quad c \leftarrow (c_k)_k,$$

where $\mathcal{G}$ is a hash function mapping into $\mathbb{Z}_p^*$.

4. For each index $i \in X$, compute

$$K_i \leftarrow r \cdot H_i, \quad (E_i, K'_i) \leftarrow \text{KEM.Enc}(pk_i)$$

5. Compute

$$E \leftarrow (E_\ell)_{\ell \in X}, \quad T \leftarrow \mathcal{H}(c, E),$$

where $\mathcal{H}$ is a hash function mapping into 2κ -bit strings.

6. For each index $i \in X$, compute

$$F_i \leftarrow S \oplus \mathcal{H}(K_i, K'_i, T)$$

7. Set

$$F \leftarrow (F_\ell)_{\ell \in X}, \quad L \leftarrow \mathcal{H}(T, F)$$

8. Compute the final encapsulated output:

$$(K, V) \leftarrow \mathcal{J}(S, L),$$

where $\mathcal{J}$ is a hash function mapping into 3κ -bit strings.

The resulting ciphertext is

$$C \leftarrow (c, E, F, V)$$

and it encapsulates the key K .

2.4.4. Decryption

Given a user's secret key usk and a ciphertext $C = (c = (c_k)_k, E = (E_\ell)_\ell, F = (F_\ell)_\ell, V)$, the decryption algorithm $\text{Dec}(\text{usk}, C)$ proceeds as follows:

1. Parse the user secret key:

$$\text{usk} = (\text{uid}, \{sk'_j\}_j), \quad \text{where } \text{uid} = (\beta_k)_k$$

2. Compute

$$T \leftarrow \mathcal{H}(c, E), \quad L \leftarrow \mathcal{H}(T, F)$$

3. For each index i such that (E_i, F_i) is in C , and for each j such that $sk'_j = (x_j, sk_j) \in \text{usk}$, do the following:

- 3.1. Run

$$K'_{i,j} \leftarrow \text{KEM.Dec}(sk_j, E_i)$$

- 3.2. Compute

$$K_j \leftarrow x_j \cdot \left(\sum_k \beta_k \cdot c_k \right), \quad S_{i,j} \leftarrow F_i \oplus \mathcal{H}(K_j, K'_{i,j}, T)$$

- 3.3. Compute

$$r' \leftarrow G(S_{i,j}), \quad (U'_{i,j}, V'_{i,j}) \leftarrow J(S_{i,j}, L)$$

- 3.4. Check whether

$$c = (r' \cdot P_k)_k \quad \text{and} \quad V'_{i,j} = V$$

- 3.5. If both checks pass, return

$$K \leftarrow U'_{i,j}$$

and terminate the algorithm.

4. If no valid pair (i, j) leads to successful decryption, return NULL.

2.5. Covercrypt and CP-ABE Use Case Example

To illustrate Covercrypt's operation, consider a simplified three-axis attribute model as demonstrated in the Covercrypt reference implementation [22]:

- Security: {None, Medium, High}
- Country: {France, Germany, UK}
- Project: {X, Y}

This configuration yields a total of $(3 + 1) \times (3 + 1) \times (2 + 1) = 48$ possible attribute combinations, referred to as *rights*. Moreover, +1 indicates the possibility of omitting one axis.

To complement the Covercrypt illustration, the operation of ciphertext-policy attribute-based encryption (CP-ABE) using the CIRCL scheme is also detailed, employing the same simplified three-axis attribute model: Security, Country, and Project.

In Covercrypt, the authority enumerates all 48 rights and invokes `generate_master_keys()`, generating a hybrid KEM key pair (classical + Kyber) for each coordinate. These subkeys collectively form the master public key (MPK) and the master secret key (MSK).

In CP-ABE, the authority generates a single set of system keys: public parameters and a master secret key.

Suppose that Alice requests access under the policy

$$\text{Security} = \text{Medium} \wedge \text{Country} = \text{France} \wedge \text{Project} = Y.$$

Upon request, Alice is issued the corresponding secret subkey for the coordinate (Medium, France, Y). Additionally, she is issued with subkeys corresponding to each subset within the set of attributes included in conjunction. Summing up, she receives the following keys:

- (Medium, France, Y),
- (–, France, Y),
- (Medium, –, Y),
- (Medium, France, –),
- (Medium, –, –),
- (–, France, –),
- (–, –, Y),
- (–, –, –).

Suppose that Carol requests access under the policy

$$\text{Security} = \text{High} \wedge \text{Country} = \text{UK}$$

Upon request, Carol is issued the corresponding secret subkey for the single coordinate (High, UK). However, because the Project dimension is not specified, Carol is also issued additional keys derived from the original policy:

$$\text{Security} = \text{High} \wedge \text{Country} = \text{UK}$$

$$\Updownarrow$$

$$\text{Security} = \text{High} \wedge \text{Country} = \text{UK} \wedge (\text{Project} = X \vee Y)$$

$$\Updownarrow$$

$$(\text{Security} = \text{High} \wedge \text{Country} = \text{UK} \wedge \text{Project} = X) \vee$$

$$(\text{Security} = \text{High} \wedge \text{Country} = \text{UK} \wedge \text{Project} = Y)$$

(High, UK, X) and (High, UK, Y). Similarly to Alice, she is also issued with subkeys corresponding to each subset within the set of attributes included in both conjunctions. Summing up, she receives the following keys:

- (High, UK, −),
- (High, UK, X),
- (High, UK, Y),
- (High, −, −),
- (High, −, X),
- (High, −, Y),
- (−, UK, −),
- (−, UK, X),
- (−, UK, Y),
- (−, −, −),
- (−, −, X),
- (−, −, Y).

In CIRCL, the authority utilizes the MSK to compute and issue one single, compact secret key: SK_{Alice} to Alice or SK_{Carol} to Carol. These are cryptographically bound to their full sets of attributes. Crucially, the size of this key is dependent on the number of attributes that they possess and not the total number of rights in the system.

Bob wishes to encrypt a message under the policy

$$\text{Security} = \text{High} \wedge \text{Country} = \text{UK} \vee \text{Country} = \text{Germany}$$

Since the Project dimension is unspecified in both conjunctions contained in the alternative, and Security is not used in the latter, Bob omits them and uses the following two rights to encrypt the symmetric key: (High, UK, −) and (−, Germany, Y). For each, he performs hybrid KEM encapsulation:

1. public key(High, UK, −) $\rightarrow (E_1, K_1)$
2. public key(−, Germany, −) $\rightarrow (E_2, K_2)$

Bob draws a single random seed S , masks it as $F_i = S \oplus K_i$, and constructs the encrypted header $\{(E_i, F_i)\}$ with a consistency tag. The payload is symmetrically encrypted under S using AES-GCM under the KEM-DEM framework.

In CP-ABE, Bob defines the access policy, which is then embedded within the ciphertext. Using the public parameters, Bob encrypts the symmetric key for the message under this policy, producing the ciphertext. The size of this ciphertext is primarily determined by the complexity of the access policy (the number of attributes and logical gates), independently of the total number of users or possible rights.

Alice, possessing only keys derived from the (Medium, France, Y) subkey, finds no matching right, and her decryption attempt fails.

In contrast, Carol holds both (High, UK, −) subkeys. She attempts decryption sequentially:

1. She applies decapsulation with (E_1, F_1) , recovers S , verifies the tag, and decrypts the payload.
2. The process halts immediately upon success, exemplifying the early abort mechanism.

For comparison, in CP-ABE, Alice similarly attempts to decrypt the CT using her secret key SK_{Alice} . Since Alice's Security: Medium attribute does not satisfy the policy requirement Security: High, she does not possess the private key corresponding to attribute Security: High and is unable to succeed during the initial pairing check.

Carol, on the other hand, possesses a key derived from the following attributes:

$$\{\text{Security: High, Country: UK, Project: Y}\},$$

so Carol’s attribute set satisfies the policy. Consequently, a single, non-iterative decryption operation succeeds, allowing Carol to recover the symmetric key and decrypt the message payload.

2.6. Formal Comparison of Security Models and Assumptions

The security of the CIRCL implementation (based on the tkn20 scheme) relies on the hardness of the Decisional Bilinear Diffie–Hellman (DBDH) problem over the BLS12-381 elliptic curve. This is a classical pairing-based assumption, which is mathematically robust but fundamentally vulnerable to Shor’s algorithm on a sufficiently powerful quantum computer.

In contrast, Covercrypt is capable of employing a hybrid security model. It layers classical elliptic curve cryptography (ECC) with Lattice-based primitives, specifically variants of the NIST-standardized ML-KEM (Kyber). The formal security of Covercrypt is maintained as long as at least one of the underlying mathematical problems (discrete logarithm or Learning With Errors—LWE) remains computationally hard.

Both schemes provide formal proofs for IND-CCA (Indistinguishability under Adaptive Chosen Ciphertext Attack) security [34,35]. Both Covercrypt and CIRCL achieve collusion resistance through the user-specific randomization of private keys. Additionally, Covercrypt integrates a traceable KEM, allowing for the forensic attribution of leaked keys—a feature not natively present in the evaluated CIRCL implementation.

CIRCL includes the access policy in the ciphertext metadata in plaintext. This allows an adversary to perform traffic analysis by inferring data sensitivity from the visible policy structure. In contrast, Covercrypt natively supports hidden access policies. The access policy is embedded within the encapsulation header, preventing unauthorized actors from learning who the intended recipients are or which attributes are required for decryption. The comparison is presented in Table 1.

Table 1. Security comparison of CIRCL and Covercrypt.

Security Metric	CIRCL (tkn20)	Covercrypt (KEMAC)
Primary Assumption	Decisional Bilinear Diffie–Hellman (DBDH) over pairings [35]	Hybrid: elliptic curve (ECC) + Lattice-based (LWE) [20,32]
Quantum Resistance	No (vulnerable to Shor’s algorithm) [30]	Yes (hybrid PQC via ML-KEM/Kyber) [32,33]
Provable Security	IND-CCA2 [35]	IND-CCA (hybrid model analysis) [34]
Policy Privacy	Public metadata (LSSS Matrix) [36]	Hidden access policies (KEMAC header) [20,32]
Identity Traceability	None (anonymity by design) [36]	Native (via TKEM mechanism) [20,34]

3. Comparison Tests with ABE Scheme

3.1. Methodology

Two libraries are compared: Covercrypt and the Cloudflare Interoperable Reusable Cryptographic Library (CIRCL), which implements ciphertext-policy attribute-based encryption (CP-ABE) on the BLS12-381 elliptic curve [23,36]. To conduct a comprehensive evaluation of the performance and practical viability of the compared libraries, a set of metrics was selected to measure both the computational overhead and the storage costs. These metrics are standard in evaluating attribute-based encryption schemes, as they di-

rectly impact scalability and efficiency in real-world applications like the Internet of Things, where resources (CPU time, memory, bandwidth) are often constrained.

To comprehensively evaluate the performance and practical utility of the compared CP-ABE libraries (Covercrypt and CIRCL), a set of metrics was selected that reflected both the computational overhead (execution time) and the storage costs (cryptographic data size). These metrics are standard in the evaluation of attribute-based encryption (ABE) schemes, as they directly impact scalability and efficiency in real-world, resource-constrained environments, particularly the Internet of Things [37–39].

3.1.1. Execution Time Metrics

In CP-ABE, computational efficiency is paramount, especially for devices with limited CPU and battery capacity [37]. The system must minimize the latency for the four fundamental cryptographic operations.

- **Master Key Generation Time:** Measures the system initialization time. Although performed only once, it is correlated with the overall setup complexity.
- **User Secret Key Generation Time:** Reflects the load on the key authority during user onboarding. This time directly impacts the scalability of user management [37].
- **Encryption Latency:** A critical metric for the data owner. It measures the computational cost of enforcing the access policy and directly influences data sharing delays. In classical CP-ABE [35], this cost is proportional to the complexity of the access policy [21].
- **Decryption Latency:** The most crucial metric for the user and constrained devices. It measures the time required to recover data, representing the heaviest computational operation in pairing-based ABE schemes. Decryption time is a major factor in energy consumption, and, in classical CP-ABE, it is highly dependent on the policy's Boolean formula size [21,38].

3.1.2. Size Metrics (Storage and Communication Overhead)

The storage and communication overhead are vital considerations in distributed systems and IoT, where bandwidth and storage are limited resources [37]. These metrics quantify the size increase of cryptographic data compared to plaintext.

- **Public Key Size:** Impacts the size of the public parameters (MPK , PP) that must be stored and accessed by all encryptors.
- **User Secret Key Size:** Affects the memory requirements of user devices and the key transmission time. The dependence of this size on the number of attributes is a key indicator of attribute scalability [39].
- **Ciphertext Size Overhead:** A crucial metric for storage providers and networks. In CP-ABE, the ciphertext embeds cryptographic elements corresponding to the access policy, meaning that its size in classical CP-ABE typically grows with the policy complexity [21,37].

3.1.3. Justification for Benchmark Parameterization

The benchmarks are parameterized as a function of the “structure size” (n), defined as the number of axes and the number of attributes in each axis. This setup is specifically chosen to induce the highest possible cost for a fixed attribute universe size in the Covercrypt system.

- Covercrypt is inherently sensitive to the number of attribute combinations (rights) necessary to represent a policy, a type of complexity that is maximized by the chosen axis–attribute structure.

- Classical ABE (CIRCL) performance is primarily influenced by the Boolean access formula size, which is also related to the growth of n .

The objective of this dual parameterization is to create a worst-case testing scenario that maximally stresses both schemes as a function of n , enabling the characterization of the thresholds at which the cost of Covercrypt's construction exceeds that of classical ABE.

For traditional pairing-based ABE (e.g., BLS12-381), performance is primarily a function of the policy's Boolean formula size. CIRCL provides optimized routines and supports pairing operations such as multi-scalar multiplications and Miller loops [23,36]. In contrast, Covercrypt's performance is sensitive to both the overall attribute universe and the specific policy shape—since it must expand to multiple attribute combinations (rights).

Consequently, our benchmark studies examined the execution time and storage overhead as functions of two orthogonal parameters: the size of the defined attribute universe and the complexity of the access policy. Our objective was to characterize the thresholds at which Covercrypt's cost exceeds that of classical ABE. Given the possible complexity of the attribute distribution between axes, benchmarks were executed in relation to n , where n is both the number of axes and number of attributes in each axes, resulting in an amount of generated rights for a fixed attribute number that was sufficient for effective analysis. n is later related to the “structure size”. Further analysis for different structures, where the number of axes and attributes differs, is possible, but it was not conducted in this work.

Moreover, Covercrypt was tested in two modes:

1. *Classical*, using only classical KEM primitives;
2. *Post-quantum*, using a hybrid KEM setup involving Kyber.

These modes are compared directly in the subsequent discussion of the results.

All experiments were conducted on a desktop system with the specifications presented in Table 2.

Table 2. Desktop system specifications used for conducting experiments.

Component	Description
Processor	Intel i5-13600KF @ 3.5 GHz
RAM	32 GB
Operating system	Ubuntu 24.04 WSL (Windows 11 host)
CIRCL library version	CIRCL v1.6.1
Covercrypt library version	Covercrypt v15.0.0

3.2. Master Key Generation Time

The master key generation algorithm in Covercrypt produces a distinct key pair for each admissible right. The cardinality of the set of rights R within the space Ω is equal to the product of all axes' sizes plus 1:

$$|R| = \left(\prod_{i=0}^m |A_i| + 1 \right), \quad (1)$$

where A_i denotes the set of attributes defined on axis i , and m is the total number of axes. The term +1 accounts for the possibility of omitting a particular axis. Under the parameterization adopted in this benchmark, this simplifies to

$$|R| = \left(\prod_{i=0}^n n + 1 \right) = (n + 1)^n. \quad (2)$$

Table 3 reports the measured master key generation times, and the corresponding visualization is provided in Figure 1. The empirical results align with the theoretical expression above, exhibiting an exponential growth pattern with respect to the structure size. For comparison, in CIRCL, the master key generation time is independent of the structure size, remaining approximately constant at $\sim 6000 \mu\text{s}$ in our benchmark.

Table 3. Master key generation time in relation to Ω size for Covercrypt with classical and post-quantum approaches.

Structure Size	Master Key Generation Time (Classical, μs)	Master Key Generation Time (Post-Quantum, μs)
1	12,827	25,684
2	57,152	66,579
3	398,813	480,034
4	3,918,226	4,680,156
5	49,437,580	58,380,727
6	749,772,390	888,053,063

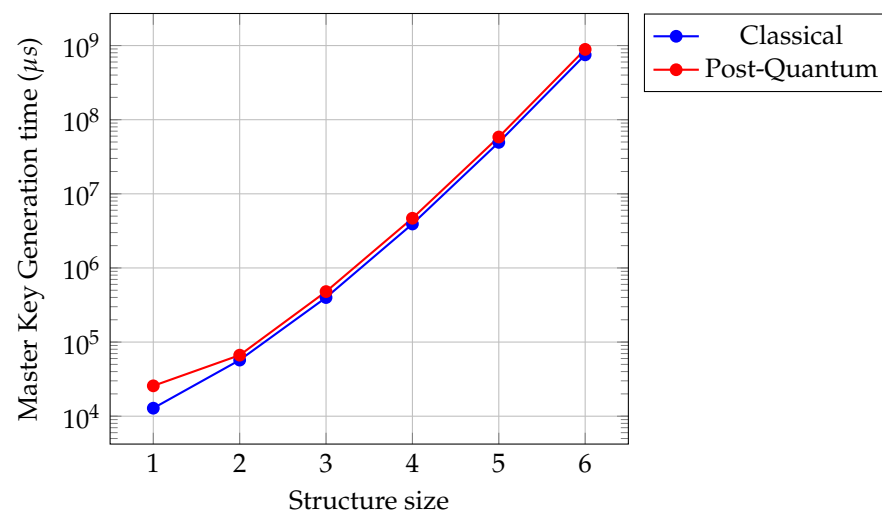


Figure 1. Master key generation time in relation to Ω size for Covercrypt with classical and post-quantum approaches.

3.3. Master Key Size

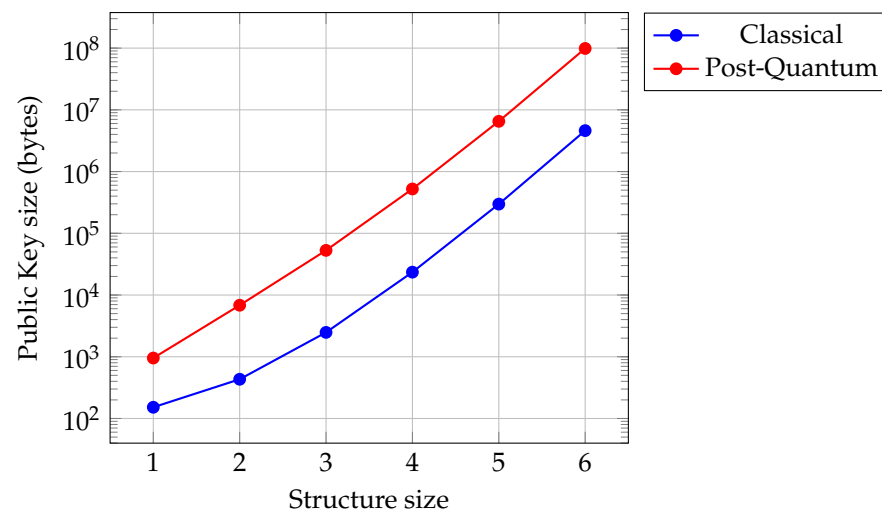
During master key generation, only one master secret key is generated. However, the public key consists of many keys that are bound to each right. Similarly to the master key generation time, the public key size depends on the number of possible rights defined in space Ω . In this benchmark, it is equal to

$$|R| = \left(\prod_{i=0}^n n + 1 \right) = (n + 1)^n. \quad (3)$$

Table 4 shows the public key sizes generated in the process, which are also visualized with plots in Figure 2. The results are in line with the number given above, indicating a quadratic relation with the structure size. For comparison, the public key size in CIRCL is always exactly 3282 bytes.

Table 4. Master key generation size in relation to Ω size for Covercrypt with classical and post-quantum approaches.

Structure Size	Public Key Size (Classical, Bytes)	Public Key Size (Post-Quantum, Bytes)
1	152	953
2	432	6833
3	2481	52,882
4	23,472	522,675
5	297,084	6,517,084
6	4,605,513	98,723,913

**Figure 2.** Public key size in relation to Ω size for Covercrypt with classical and post-quantum approaches.

3.4. User Key Generation Time

As secret keys connected to rights are generated during the setup phase, these are prescribed for the user from the master secret key. The key-assigning entity must, however, determine which rights the user's policy expands into. After this, the steps described in Section 2.4.2 occur, including the preparation of the user's secret identity and updating both the tracing and master secret keys. The user secret key is also signed, which takes most of the time in this step and is proportional to the generated user key size. Table 5 presents the generation time for user keys with respect to the size of the structure and the length of the user policy (number of defined attributes) for the classical and post-quantum approaches for a policy consisting of one conjunction, and Table 6 presents the results for an extremely contrasting policy consisting of one alternative. The difference between the policies' construction results in different numbers of rights expanded and therefore different key generation times. This is explained in Section 3.5. The results for each of the variants are visualized in plots in Figures 3–6. Notice that the time for the post-quantum approach is increased greatly, mostly due to the larger subkeys.

Table 5. Time of user key generation with respect to structure and user policy sizes for classical and post-quantum approaches for conjunction.

Structure Size	User Policy Attributes	Time (Classical Conj., μ s)	Time (PQ Conj., μ s)
1	1	244	622
2	1	265	2201
	2	251	1656
3	1	506	12,406
	2	360	6336
	3	308	3097
4	1	2279	99,779
	2	1009	41,746
	3	554	16,071
	4	350	6118
5	1	23,830	1,028,325
	2	8325	342,224
	3	2706	113,578
	4	1051	37,730
	5	540	12,560
6	1	326,474	13,475,799
	2	90,537	3,823,178
	3	24,705	1,108,539
	4	7446	313,978
	5	2247	88,200
	6	1010	25,399

Table 6. Time of user key generation with respect to structure and user policy sizes for classical and post-quantum approaches for alternative.

Structure Size	User Policy Attributes	Time (Classical Alt., μ s)	Time (PQ Alt., μ s)
1	1	256	704
2	1	285	2367
	2	308	3231
3	1	548	13,230
	2	681	19,382
	3	831	22,679
4	1	2360	104,845
	2	4351	163,890
	3	5008	202,349
	4	5801	225,374
5	1	23,804	1,051,628
	2	41,139	1,730,335
	3	55,445	2,170,905
	4	65,095	2,553,004
	5	76,653	2,748,792
6	1	332,138	13,586,972
	2	621,244	23,135,262
	3	835,961	30,117,274
	4	985,815	35,578,373
	5	1,194,363	38,712,912
	6	1,311,403	41,207,026

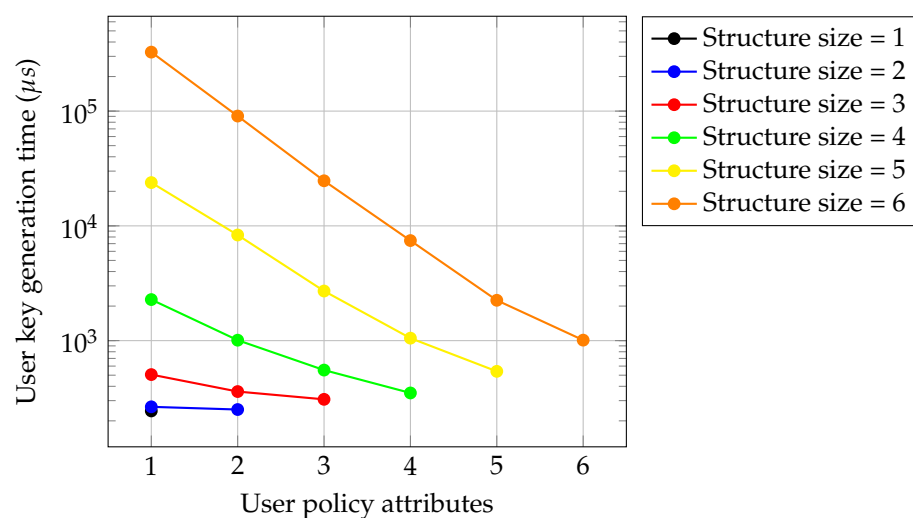


Figure 3. User key generation time in relation to user policy length and structure size for Covercrypt with classical approach for conjunction.

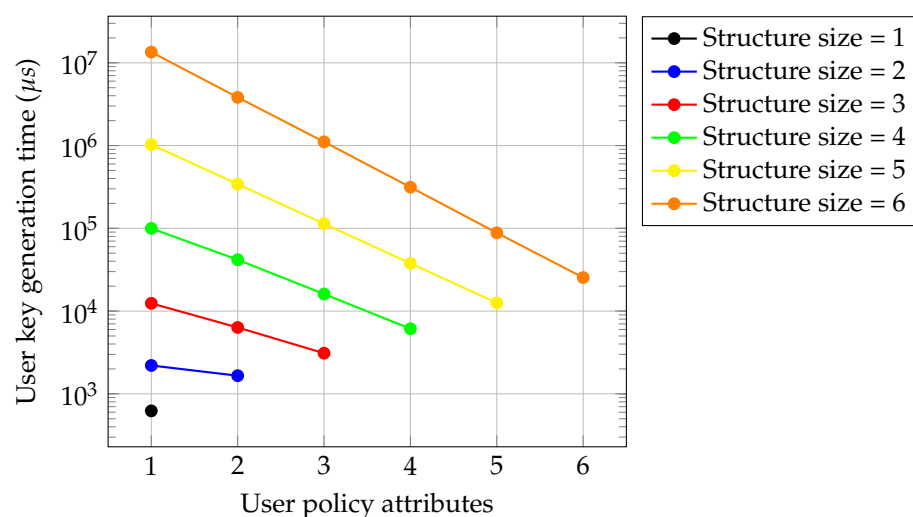


Figure 4. User key generation time in relation to user policy length and structure size for Covercrypt with post-quantum approach for conjunction.

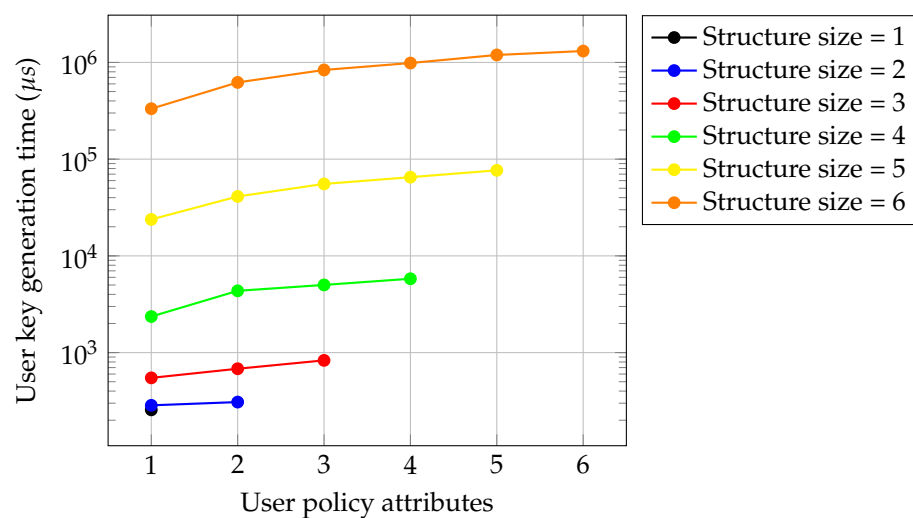


Figure 5. User key generation time in relation to user policy length and structure size for Covercrypt with classical approach for alternative.

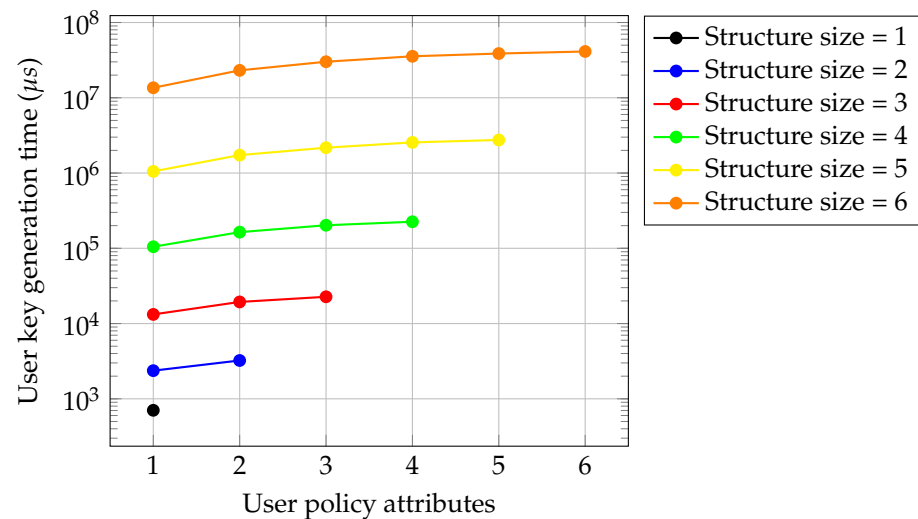


Figure 6. User key generation time in relation to user policy length and structure size for Covercrypt with post-quantum approach for alternative.

In CIRCL, user Key generation is performed by generating a private key corresponding to each attribute in the policy. Additionally, the key-generating entity creates a randomized secret bound to the user for collusion resistance purposes and applies it to each private key. The time of key generation for different policy lengths, corresponding to Covercrypt, in CIRCL is presented in Table 7. Unlike in Covercrypt, the user key generation time does not depend on the policy form—it is very similar for both a single conjunction and a single alternative. A comparison of both libraries for classical and post-quantum Covercrypt and CIRCL is presented in Figures 7 and 8.

Table 7. Private key generation time in relation to policy length in CIRCL.

Policy Length	User Key Generation Time (μs)—Conjunction	User Key Generation Time (μs)—Alternative
1	19,955	19,988
2	27,179	27,226
3	34,855	34,386
4	40,798	41,536
5	49,760	48,036
6	55,603	54,899

3.5. User Key Size

As described in Section 3.4, the secret keys associated with rights are derived during the setup phase from the master secret key, once the key-issuing authority determines the set of rights implied by a given user policy. In this benchmark, the cardinality of the rights set is given by

$$|R| = (n + 1)^n.$$

The number of rights assigned to a user is determined by the number of attributes specified in the policy. Consider policies consisting of a single clause in disjunctive normal form (DNF), where all attributes appear in conjunction.

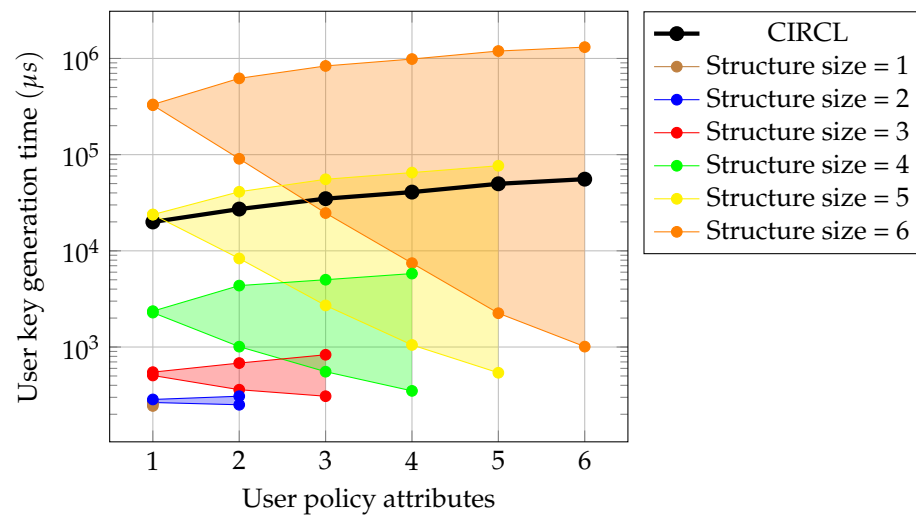


Figure 7. User key generation time in relation to user policy length and structure size—comparison for best and worst cases of classical Covercrypt and CIRCL.

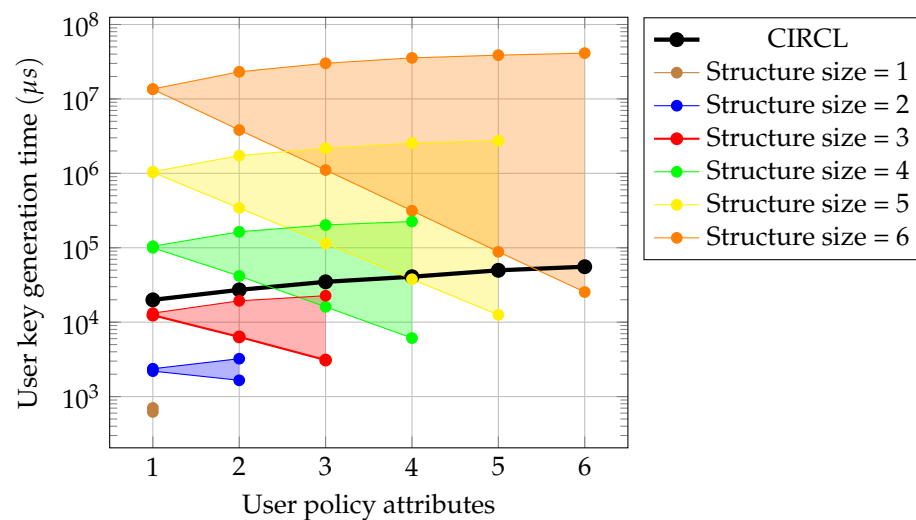


Figure 8. User key generation time in relation to user policy length and structure size—comparison for best and worst cases of post-quantum Covercrypt and CIRCL.

The policy expansion is defined as follows. Let the policy be expressed as a conjunction of l attributes, denoted by

$$P = \{a_1 \wedge a_2 \wedge \dots \wedge a_l\}.$$

A family of subsets is constructed

$$\mathcal{S} = \{S \subseteq P \mid |S| \leq l\},$$

which enumerates all subsets of P of size 0 up to l . The expanded rights set is then obtained as the Cartesian product,

$$R(P) = \mathcal{S} \times \prod_{a \in \mathcal{A} \setminus \mathcal{A}_P} D_a,$$

where $\mathcal{A}$ denotes the set of axes, $\mathcal{A}_P$ denotes the set of axes that contain attributes from P , and D_A is the set of attributes in axis A . The total number of rights assigned to the user with policy P in relation to n and l is equal to

$$|R(P)| = |S| \cdot \prod_{a \in \mathcal{A} \setminus \mathcal{A}_P} |D_a| = 2^l \cdot \prod_{i=1}^{n-l} n = 2^l \cdot (n+1)^{n-l}. \quad (4)$$

The situation differs when considering a contrasting approach with policies consisting of alternatives only. Then, policy expansion is defined as follows. Let the policy be expressed as a conjunction of l attributes, denoted by

$$P = \{a_1 \vee a_2 \vee \dots \vee a_l\}.$$

In l different sets Z_i (axes), one can choose a fixed a_i attribute. The expanded rights set is then obtained as the Cartesian product for each of the fixed attributes:

$$R(a_i) = a_i \times \prod_{D \in \mathcal{A} \setminus \mathcal{A}_{a_i}} (D \cup \emptyset),$$

where $\mathcal{A}$ denotes the set of axes, $\mathcal{A}_{a_i}$ denotes the axis that contains the attribute a_i , and D is an axis not containing a_i . Note that D is summed with $\emptyset$ as attributes from different dimensions cannot be set. All rights generated for the policy can then be described as

$$R(P) = \bigcup_{i=1}^l R(a_i).$$

To determine the cardinality of $R(P)$ in this case, the size of a union of sets must be calculated. Since the sets $R(a_i)$ are not disjoint (e.g., the tuple with a_1 and a_2 is in both $R(a_1)$ and $R(a_2)$), one cannot simply sum their individual sizes.

The most straightforward method is to calculate the size of the complement $R(P)^C$ and subtract it from the total universe U .

Let U be the universe of all possible rights, defined as the Cartesian product of all n axes, where each axis D_i is extended with the $\emptyset$ element.

$$U = \prod_{i=1}^n (D_i \cup \{\emptyset\})$$

Assume that there are n total axes, and each axis D_i has cardinality of n . Therefore, the cardinality of each extended axis is $|D_i \cup \{\emptyset\}| = n + 1$. The total size of the universe is

$$|U| = (n+1)^n$$

The set $R(P)$ contains all tuples where at least one specified attribute a_i (for $i = 1, \dots, l$) is present at its corresponding axis.

The complement, $R(P)^C$, therefore contains all tuples where *none* of the specified attributes a_i are present. A tuple $(x_1, \dots, x_n)$ is in $R(P)^C$ if and only if $x_i \neq a_i$ for all $i \in \{1, \dots, l\}$.

The cardinality of $R(P)^C$ can be found by counting the available choices for each position in the tuple.

- For the l axes ($i = 1, \dots, l$) involved in the policy P : The element x_i can be any attribute from D_i *except* the specified a_i . This leaves $n - 1$ choices for each of these l axes. The

element $\{\emptyset\}$ is not considered as the policy is always expanded from a_i to $\{\emptyset\}$ within the dimension.

- For the remaining $n - l$ axes ($j = l + 1, \dots, n$) not involved in the policy: The element x_j can be any attribute from $D_j \cup \{\emptyset\}$. This gives $n + 1$ choices for each of these $n - l$ axes.

The total size of the complement is the product of these choices:

$$|R(P)^C| = \underbrace{((n-1) \cdot (n-1) \cdots (n-1))}_{l \text{ times}} \cdot \underbrace{((n+1) \cdot (n+1) \cdots (n+1))}_{n-l \text{ times}} = (n-1)^l \cdot (n+1)^{n-l}$$

Finally, the total number of rights $|R(P)|$ is the size of the universe minus the size of the complement:

$$|R(P)| = |U| - |R(P)^C| = (n+1)^n - (n-1)^l (n+1)^{n-l} \quad (5)$$

This can also be expressed in factored form:

$$|R(P)| = (n+1)^{n-l} \left[(n+1)^l - (n-1)^l \right] \quad (6)$$

User key sizes are presented in Tables 8 and 9 for the conjunction and alternative scenarios, respectively. For comparison, in CIRCL, the user receives only one private key related to the policy that he is given; however, it consists of subkeys connected with each of the attributes. Thus, it depends entirely on the number of attributes that the policy consists of. The sizes of the user keys for the corresponding numbers of attributes are presented in Table 10. Unlike in Covercrypt, the user key size does not depend on the policy form—it is the same for both a single conjunction and a single alternative.

Table 8. User secret key (USK) size in bytes with respect to structure and user policy sizes for classical and post-quantum approaches for conjunction structure.

Structure Size	User Policy Attributes	User Key Size (Classical Conj., Bytes)	User Key Size (PQ Conj., Bytes)
1	1	234	1866
2	1	380	8540
	2	307	5203
3	1	1347	51,939
	2	751	25,231
	3	455	11,879
4	1	9639	416,007
	2	3923	165,491
	3	1655	65,303
	4	755	25,235
5	1	100,820	4,329,332
	2	33,428	1,441,844
	3	11,156	479,540
	4	3795	158,835
	5	1363	51,955
6	1	1,337,522	56,193,938
	2	378,836	16,050,932
	3	107,376	4,583,952
	4	30,516	1,308,372
	5	8756	372,692
	6	2595	105,411

Table 9. User key size in bytes with respect to structure and user policy sizes for classical and post-quantum approaches for alternative.

Structure Size	User Policy Attributes	User Key Size (Classical Alt., Bytes)	User Key Size (PQ Alt., Bytes)
1	1	234	1866
2	1	380	8540
	2	453	11,877
3	1	1347	51,939
	2	1943	78,647
	3	2243	92,003
4	1	9639	416,007
	2	15,354	666,522
	3	18,801	816,849
	4	20,880	907,056
5	1	100,820	4,329,332
	2	168,212	7,216,820
	3	213,332	9,142,004
	4	243,540	10,425,588
	5	263,764	11,281,396
6	1	1,337,522	56,193,938
	2	2,296,207	96,336,943
	3	2,983,432	125,012,968
	4	3,476,057	145,497,593
	5	3,829,182	160,130,718
	6	4,082,307	170,583,843

Table 10. Private key sizes in relation to policy length in CIRCL.

Policy Length	User Key Size (Bytes)—Conjunction	User Key Size (Bytes)—Alternative
1	2347	2347
2	2778	2778
3	3209	3209
4	3640	3640
5	4071	4071
6	4502	4502

3.6. Encryption Time

In Covercrypt, only one key is used for encryption for policies in the form of a single conjunction. As described in Section 2.4.3, essentially, a symmetric key is generated for the ciphertext and it is then encrypted with a pair of keys corresponding to the policy. If the policy contains more conjunctions in DNF form, then the symmetric key is encrypted for each, as shown in Table 11 for the two extreme opposite cases of a conjunction only and alternative only.

In CIRCL, the situation is more complex, as encryption is based upon an LSSS mechanism, as described in Section 2.3.3. The encryption time depends on the number of attributes contained in the ciphertext policy, as presented in Table 11.

Table 11. Encryption time in relation to policy length in CIRCL and Covercrypt.

Policy Length	Encryption Time in CIRCL (μ s)	Encryption Time in Covercrypt Classic (Conj., μ s)	Encryption Time in Covercrypt Post-Quantum (Conj., μ s)	Encryption Time in Covercrypt Classic (Alt., μ s)	Encryption Time in Covercrypt Post-Quantum (Alt., μ s)
1	14,884	18,639	20,913	20,177	21,910
2	18,030	18,510	20,928	26,402	31,836
3	21,469	18,701	20,818	33,957	37,297
4	24,936	18,851	20,770	39,970	44,413
5	28,731	18,835	20,802	46,155	53,799
6	32,310	18,837	20,597	52,858	61,747

3.7. Encryption Overhead

For both libraries, the encryption overhead is not connected with the size of the ciphertext, as the scheme's purpose is key encapsulation. In Covercrypt, each overhead for a fixed symmetric key size is constant, and, in our case (AES-128), it is exactly 32 bytes for classical and 800 bytes for post-quantum mode. Similarly to the encryption time, when the policy contains more conjunctions in DNF form, then the symmetric key is encrypted for each, as shown in Table 12 for the two extreme opposite cases of a conjunction only and alternative only. In CIRCL, the size grows with the policy length due to components C_i , as explained in Section 2.3.3. Table 12 summarizes the results.

Table 12. Encryption overhead in relation to policy length in CIRCL and Covercrypt.

Policy Length	Encryption Overhead in CIRCL (Bytes)	Encryption Overhead in Covercrypt Classic (Conj., Bytes)	Encryption Overhead in Covercrypt Post-Quantum (Conj., Bytes)	Encryption Overhead in Covercrypt Classic (Alt., Bytes)	Encryption Overhead in Covercrypt Post-Quantum (Alt., Bytes)
1	2343	115	883	115	883
2	2691	115	883	147	1683
3	3039	115	883	179	2483
4	3387	115	883	211	3283
5	3735	115	883	243	4083
6	4083	115	883	275	4883

3.8. Decryption Time

In Covercrypt, the policies embedded within ciphertexts are hidden. Although only a single key is ultimately required to decrypt the message, the recipient does not know a priori which key to use. Consequently, the receiver attempts to decrypt the message sequentially with each of their possessed keys, as detailed in Section 2.4.4.

The Covercrypt library implementation [22] iterates through all available decryption keys and then, in an inner loop, through all available key encapsulations. Since the order of both keys and encapsulations is randomized before the loops commence, the resulting expected computation time is also randomized.

Each key decapsulation involves two sequential steps: the check phase and the final decapsulation phase, as outlined in Section 2.3.4. The approximate time values for these phases, based on this benchmark, are approximately 6450 μ s for the check phase and

12,726 μs for the final decapsulation phase for classical. For post-quantum, the values are 8192 μs and 12,859 μs , respectively. The approach for ciphertext decapsulation requires the user to check each of their issued rights' keys against every encapsulated key. The Expected Value of Checks (EV_{checks}) required before finding the first fitting key is given by the following expression, derived from the randomized selection process:

$$\begin{aligned} EV_{\text{checks}} &= |R(P)| \cdot EV_{\text{checked_wrong_encapsulations}} + EV_{\text{checked_rights_with_correct_key}} \\ &= |R(P)| \cdot \left(\frac{n_{\text{enc}} + 1}{L + 1} - 1 \right) + \frac{|R(P)| + 1}{2} \\ &= |R(P)| \cdot \left(\frac{n_{\text{enc}} - L}{L + 1} \right) + \frac{|R(P)| + 1}{2}, \end{aligned}$$

where

- $|R(P)|$ is the number of rights (keys) that the user possesses that are not the correct key for decapsulation;
- n_{enc} is the total number of key encapsulations attached in the ciphertext;
- L is the number of keys that the user possesses that can be used to decapsulate any of the available encapsulations.

The total decryption time can thus be estimated by multiplying the EV_{checks} value by the average time taken for the check phase. Example time values obtained from the benchmark, presented as an average of 100 runs for each policy length under the condition $n_{\text{enc}} = L$, are detailed in Table 13.

In contrast to the iterative approach of Covercrypt, the CIRCL scheme requires the recipient to first verify whether their attributes satisfy the ciphertext policy. Upon successful verification, the decryption attempt proceeds, typically resulting in only one successful decryption operation.

Since the duration of the CIRCL decryption process is dependent on the cryptographic operations related to the policy complexity (i.e., the number of attributes or gates involved), its duration is primarily determined by the policy length. The decryption results for CIRCL are also presented in Table 13.

Table 13. Example decryption times in relation to policy length in CIRCL and Covercrypt.

Policy Length	Decryption Time in CIRCL (μs)	Decryption Time in Covercrypt Classic (μs)—Conjunction	Decryption Time in Covercrypt Post-Quantum (μs)—Conjunction	Decryption Time in Covercrypt Classic (μs)—Alternative	Decryption Time in Covercrypt Post-Quantum (μs)—Alternative
1	5640	35,670	33,757	36,220	33,834
2	5810	40,614	41,575	52,650	57,114
3	6545	52,222	56,449	201,293	266,168
4	6213	81,956	85,339	1,630,697	2,207,979

3.9. Analysis of ABE Scheme Scalability in IoT

To determine the optimal cryptographic scheme for a specific Internet of Things (IoT) environment, a theoretical analysis of the complexity and scalability concerning key parameters is crucial. Both ciphertext-policy attribute-based encryption (CP-ABE, as implemented by CIRCL) and the key encapsulation mechanism with access control (KEMAC, as implemented by Covercrypt) must be evaluated based on their operational costs as a function of the system parameters.

3.9.1. Complexity Comparison: CIRCL vs. Covercrypt

The performance of attribute-based encryption schemes is typically measured by their computational and communication complexity, which depend on

- The total number of attributes in the system, often represented as the size of the attribute universe N or the number of attribute axes m ;
- The number of attributes in a user's secret key, $|A_U|$;
- The complexity of the access policy $|\mathbb{A}|$ (e.g., number of logic gates in the linear secret sharing scheme (LSSS) matrix for CIRCL or the resulting number of rights for Covercrypt).

Brief analysis is presented in Table 14.

Table 14. Theoretical complexity comparison of CP-ABE (CIRCL) and KEMAC (Covercrypt).

Operation Metric	CIRCL (Pairing-Based CP-ABE)	Covercrypt (Hybrid KEMAC)	Implications for Scalability
Master Key Setup Time	Constant $O(1)$	Exponential $O(\Omega)$, where $ \Omega = \prod(\text{Axis Size})$	CIRCL is vastly superior for large, high-dimensional attribute spaces.
Master Key Size (MPK)	Constant (small) $O(1)$	Exponential $O(\Omega)$	Covercrypt is impractical for centralized key authorities with many axes.
User Key Gen. Time (KeyGen)	Linear $O(A_U)$	Exponential $O(SK_U)$ due to policy expansion into rights	CIRCL scales better with the number of attributes assigned to a user.
Ciphertext Size ($ C $)	Linear $O(\mathbb{A})$	Constant $O(1)$ (dependent only on KEM size)	Covercrypt minimizes communication overhead, crucial for low-bandwidth links.
Decryption Time	Linear $O(\mathbb{A})$ (one pairing computation)	Linear $O(SK_U)$ (iterative KEM decapsulation + early abort)	CIRCL is typically faster as it performs a single, optimized pairing operation.

3.9.2. Inferred Scalability Thresholds

The analysis, combined with the raw benchmark results from the previous sections, establishes clear operational thresholds.

1. **Setup and MPK Size Threshold:** Covercrypt's exponential cost in the Setup phase and the resulting size of the MPK make it unsuitable for systems where the attribute universe (Ω) is large or dynamic (e.g., $|\Omega| > 100$). This is the primary constraint.
2. **Communication Threshold:** Covercrypt's constant and small ciphertext overhead (especially ≈ 883 bytes for post-quantum mode for one conjunction policy) gives it a definitive advantage in environments defined by extreme bandwidth constraints.
3. **Decryption Speed Threshold:** CIRCL's faster decryption time (around $6633 \mu\text{s}$ vs. $20,034 \mu\text{s}$ for Covercrypt PQ) makes it preferable where low-latency data access is critical for the user experience, provided that the policy is not overly complex.

Based on these thresholds, the schemes are best suited for distinct military IoT operational environments. The decryption latency is the primary driver of energy consumption in battery-powered IoT nodes. The 300% faster decryption throughput demonstrated by

CIRCL reduces the active CPU time per transaction, which is vital for the longevity of unattended ground sensors and soldier-worn devices. Conversely, the higher computational cost of Covercrypt PQ must be viewed as an unavoidable “security tax” to achieve quantum resilience in high-threat edge scenarios.

3.10. Special IoT Application Use Cases and Comparative Scenario Analysis

Two distinct special IoT operational scenarios to illustrate the trade-offs between the highly scalable pairing-based CP-ABE (CIRCL) and the bandwidth-efficient, quantum-safe KEMAC (Covercrypt) are defined.

3.10.1. Scenario I: Enterprise/Federated Network Access (CORP-NET)

This scenario represents the high-scale corporate and command-and-control backbone, characterized by large user bases, distributed trust domains, and standard high-throughput communication links.

- Environment: Headquarters, military bases, high-capacity regional networks, and federated networks with allied nations.
- Scale: $\sim 50,000$ users/devices (servers, workstations, smart devices).
- Attribute Space (Ω): Large. Example: 5 attribute axes, each with 5 values (Role, SecurityClearance, Department, Nationality, MissionStatus). Total Rights: $|\Omega| = (5 + 1)^5 = 7776$.
- Critical Requirements: Scalability and Setup Efficiency. The key authority must be able to manage a large, frequently changing attribute universe without prohibitive time/storage costs. A minimal user key size ($|SK_U|$) is required for mobile devices.

Suitability of the libraries for this scenario is presented in Table 15.

Table 15. CIRCL and Covercrypt comparison for Scenario I.

Metric	CIRCL	Covercrypt (PQ)	Suitability for Scenario I
Setup Cost ($ \Omega = 7776$)	~ 6 ms ($O(1)$)	~ 53 s (exponential $O(\Omega)$)	CIRCL is essential. Covercrypt’s 53 s setup time and resulting ~ 6.5 MB public key is already challenging for a dynamic system.
User Key Size (5 attributes)	~ 4 KB (small)	Depending on attribute number and policy construction, between 50 KB and 160 MB $\approx$ large	CIRCL is preferred. Easier distribution and storage at thousands of endpoints.
Decryption Time	~ 6.6 ms (fast)	~ 20 ms (slower)	CIRCL is preferred. Ensures low-latency access to enterprise resources for a large user base.

Conclusions for CORP-NET (Scenario I): CIRCL is the superior choice. Its fast, constant-time Setup and compact key sizes ensure the necessary scalability and manageability required for a large, enterprise-grade system, outweighing the benefits of Covercrypt’s post-quantum readiness in this specific context.

3.10.2. Scenario II: Tactical Edge Network (TEN)

This scenario represents the low-bandwidth, mission-critical networks at the extreme tactical edge, such as infantry platoons or vehicle formations. It is defined by severe communication constraints and the need for long-term security.

- Environment: Dismounted soldiers, armored vehicles, tactical drones, and field command posts. Communication via constrained radio or satellite links.
 - Scale: ~500 devices/users (small, isolated groups).
 - Attribute Space (Ω): Small and Narrow. Example: 3 attribute axes, each with 3 values (Unit Type, Command Level, Classification). Total Rights: $|\Omega| = (3 + 1)^3 = 64$.
 - Critical Requirements: Minimal Communication Overhead and Post-Quantum Security.
- Suitability of the libraries for this scenario is presented in Table 16.

Table 16. CIRCL and Covercrypt comparison for Scenario II.

Metric	CIRCL	Covercrypt (PQ)	Suitability for Scenario II
Setup Cost ($ \Omega = 64$)	~6 ms	~0.5 s (acceptable)	Both acceptable.
Ciphertext Overhead (6 attributes)	~4083 B (growing)	~883 B (constant)	Covercrypt is preferred. Reduced data transfer (over 4× less overhead) is vital for low-bandwidth radio links.
Security Model	Classical ABE	Hybrid post-quantum	Covercrypt is preferred. Essential for protecting long-term military intelligence against quantum threats.
Policy Privacy	Public	Hidden	Covercrypt is preferred. Prevents adversaries from inferring data sensitivity by reading the policy tag.

Conclusions for TEN (Scenario II): Covercrypt is the preferred choice. While its setup is slower than CIRCL’s, the absolute time is acceptable for a small attribute universe (<1 s). The constant, minimal ciphertext overhead and the inherent post-quantum security are paramount for constrained, high-threat tactical environments, making it the more effective cryptographic enabler for data-centric security at the edge. The measured ciphertext overhead of 883 bytes for Covercrypt PQ, compared to over 4000 bytes for CIRCL with six attributes, represents a critical operational advantage. In tactical environments utilizing narrowband IoT or legacy VHF/UHF radio links, where data rates are extremely limited, this four-times reduction directly improves the packet delivery ratios and reduces the probability of interception by shortening the transmission window.

4. Conclusions

This comparative performance analysis between the pairing-based CP-ABE scheme CIRCL and the hybrid post-quantum KEMAC Covercrypt provides critical insights for implementing data-centric security (DCS) across the diverse operational spectrum of military Internet of Things (IoT) environments. Our findings affirm that no single scheme offers a universal solution; rather, the optimal cryptographic choice is dictated by scenario-specific constraints related to scale, bandwidth, and security horizons. The quantitative findings reveal a critical dichotomy in the architectural suitability of these schemes. The pairing-based CIRCL implementation demonstrates superior scalability in dynamic environments,

characterized by a constant-time master key setup phase ($O(1)$) and linear scalability with respect to user attributes. Conversely, the Covercrypt scheme, while offering robust post-quantum security and a constant ciphertext size—a decisive advantage for low-bandwidth tactical links—suffers from exponential setup complexity ($O(|\Omega|)$) relative to the attribute universe size. This exponential cost renders Covercrypt operationally prohibitive for high-dimensional attribute spaces typical of large-scale coalition networks but ideal for the static, defined ontologies of tactical edge deployments. Node mobility and the “dynamic join-and-leave” behavior of tactical units require high agility from the key authority. CIRCL’s constant-time setup ($O(1)$) and efficient key issuance allow for the immediate onboarding of new coalition partners or retasked units. Covercrypt’s structural rigidity, caused by the exponential cost of modifying the attribute universe ($O(|\Omega|)$), may limit its use in highly fluid scenarios where new mission-specific attributes must be introduced frequently. This report concludes that no single cryptographic primitive suffices for the entire spectrum of special IoT applications. Instead, a split-horizon deployment strategy is recommended: utilizing pairing-based CP-ABE for the scalable, dynamic operational core and deploying hybrid KEMAC solutions at the tactical edge, where bandwidth efficiency and quantum resilience are paramount. This synthesis provides a roadmap for network planners to balance the competing demands of computational latency, communication overhead, and long-term cryptographic assurance.

The fundamental performance disparity arises from the architectural differences in managing the attribute universe (Ω):

- CIRCL exhibits superior scalability and agility for the key authority. Its constant $O(1)$ complexity for the Setup phase and compact public parameters (PP) are essential in supporting a high-dimensional or rapidly evolving attribute space. Crucially, CIRCL’s architecture is inherently flexible, accommodating the frequent addition of new attributes in enterprise scenarios without requiring the costly and time-consuming recalculation of the central MPK.
- Covercrypt’s exponential $O(|\Omega|)$ complexity for Setup is a severe bottleneck for large deployments, resulting in an unwieldy MPK (e.g., ~ 98 MB for structure size $n = 6$). Furthermore, any change to the attribute universe—even adding a single new attribute or axis—necessitates the complete regeneration and distribution of the MPK, rendering it structurally rigid for broad enterprise adoption.

The data imply a rigid operational doctrine for Covercrypt. The exponential cost of setup means that attributes cannot be treated as tags. In a CIRCL system, a user might tag a document with “Date: 2025-11-22” and “Coordinates: 34.55,12.33”. In Covercrypt, creating attributes for every date or coordinate is impossible as it would explode the universe size. Covercrypt enforces coarser granularity in access control. Access must be defined by broad categories (Role, Region) rather than fine-grained metadata. The tsk (tracing secret key) mechanism in Covercrypt offers an operational advantage. In standard CP-ABE, keys are often shared anonymously. In Covercrypt, the mathematical embedding of the user identity within the key encapsulation allows for forensic attribution. This feature is critical for zero-trust architectures. It moves the trust boundary from “possession of the device” to “accountability of the actor”. Covercrypt natively supports hidden access policies. The ciphertext does not reveal who is allowed to read it (unless they can decrypt it). In contrast, standard CP-ABE (CIRCL) often includes the policy structure in plaintext (metadata) to allow the decryption algorithm to function. Hiding the policy prevents traffic analysis. An adversary capturing encrypted packets cannot distinguish between a “general broadcast” and a “top secret strike order”, preventing them from inferring imminent operations based on the target audience of the messages. It should be noted that a hidden access policy feature can be achieved in CP-ABE scenarios by using homomorphic encryption. In terms of

runtime metrics, CIRCL demonstrated decryption throughput that was roughly 300% faster than in Covercrypt's post-quantum mode, making it the clear choice for latency-sensitive applications. Conversely, Covercrypt's design provides a significant advantage in terms of communication overhead, achieving a constant, minimal ciphertext size (~883 bytes for PQ mode) compared to the growing, higher overhead of CIRCL (~4083 bytes for 6 attributes).

The analysis leads to distinct recommendations for the two scenarios.

- **CIRCL for Scalability and Real-Time Control (CORP-NET):** The classical pairing-based approach is optimal for high-scale, corporate, or federated networks (Scenario I). Its fast key generation and high decryption throughput are paramount in managing a large user base and ensuring a responsive experience. While lacking native post-quantum security (PQS), this vulnerability can be successfully mitigated by adopting PQS protocols for data transit within the high-bandwidth network backbone, effectively layering security without sacrificing performance.
- **Covercrypt for Communication Efficiency and PQS at the Edge (TEN):** The KEMAC-based approach is ideally suited for tactical edge networks (Scenario II). The constant, minimal ciphertext overhead is a critical enabler for severely constrained radio links, offering a typical 4× reduction in data transfer compared to CIRCL. Furthermore, its native hybrid Post-Quantum capabilities and policy privacy are non-negotiable in this high-threat environment. Relying on DCS encryption as the sole cryptographic barrier is crucial here, as introducing any additional protocols would overwhelm the already constrained bandwidth.

This comparative performance analysis confirms that no single cryptographic scheme provides a universal solution for the entire spectrum of IoT environments. Instead, the choice must be dictated by specific operational constraints related to the network scale, available bandwidth, and required security horizon.

It is important to acknowledge the inherent flexibility of the CP-ABE attribute definition, exemplified by the CIRCL implementation, which provides additional operational merits even for tactical networks. The creation of flexible attribute values allows for integration with real-time operational parameters.

- **User Revocation and Future Encryption:** By concatenating attributes with precise timestamps, the system can effectively implement time-bound policies, enabling immediate user revocation (by withholding future keys) and allowing data senders to encrypt messages that can only be decrypted after a specific point in time.
- **Dynamic Policy Tagging:** Attributes can be dynamically generated based on a device's current state, such as its location (e.g., map grid coordinates) or operational status. This enables highly reactive access control, critical for emergency situations where data access must immediately align with dynamic, rapidly changing geographical contexts.

While the need for post-quantum security strongly encourages the adoption of schemes like Covercrypt, the current, prohibitive cost of its Setup phase severely restricts its practical utility in narrow, low-scale attribute universes typical of tactical deployments. For large-scale enterprise environments, CP-ABE schemes based on bilinear pairings (CIRCL) remain the more viable and agile choice until a comparably efficient post-quantum CP-ABE solution with constant-time Setup becomes widely available.

Based on the experimental results, the following guidance for system architects is proposed. For large-scale, dynamic enterprise networks, pairing-based CP-ABE is the recommended choice. Its constant-time setup and faster decryption throughput are essential in managing thousands of devices with minimal latency, outweighing the lack of native post-quantum resistance in high-bandwidth backbone environments. Conversely, for tactical edge networks characterized by severely constrained radio links and high threat levels,

the hybrid KEMAC approach is the superior enabler for data-centric security. Despite its exponential setup complexity, Covercrypt’s constant and smaller ciphertext overhead is paramount for bandwidth conservation, while its native post-quantum security and hidden access policies are non-negotiable in protecting long-term military intelligence against emerging quantum threats.

Ultimately, a split-horizon deployment strategy is recommended: utilizing the scalability of CIRCL for the operational core and the communication efficiency and quantum resilience of Covercrypt for the tactical edge. A binary choice between CIRCL and Covercrypt is not the end of the road. Several emerging frameworks offer middle-ground solutions. The current study assumes a single key authority (KA), which is a single point of failure and a bottleneck. Future research should benchmark Lewko-Waters MA-ABE [40]. This allows different authorities (e.g., US-DoD, UK-MoD, NATO-HQ) to issue attributes independently. A user can decrypt data if they possess “US-TopSecret” AND “NATO-Mission-X”, without the US and NATO sharing their master keys. This is the “holy grail” for coalition interoperability. Pure Lattice ABE schemes (e.g., BGG+) should also be investigated. These offer full quantum resistance for both the key and the policy logic but historically suffer from massive key sizes (megabytes). Benchmarking these on modern hardware would determine their viability for the considered scenarios.

Author Contributions: Conceptualization, K.K. and Z.Z.; data curation, Ł.P.; methodology, Ł.P.; resources, Ł.P.; software, Ł.P.; supervision, K.K. and Z.Z.; visualization, Ł.P. and Z.Z.; writing—original draft, Ł.P.; writing—review and editing, Ł.P., K.K. and Z.Z. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Data Availability Statement: Data are contained within the article.

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

References

1. Suri, N.; Zielinski, Z.; Tortonesi, M.; Fuchs, C.; Pradhan, M.; Wrona, K.; Furtak, J.; Vasilache, D.B.; Street, M.; Pellegrini, V.; et al. Exploiting smart city IoT for disaster recovery operations. In Proceedings of the 2018 IEEE 4th World Forum on Internet of Things (WF-IoT), Singapore, 5–8 February 2018; pp. 458–463. [\[CrossRef\]](#)
2. Munir, A.; Aved, A.; Blasch, E. Situational Awareness: Techniques, Challenges, and Prospects. *AI* **2022**, *3*, 55–77. [\[CrossRef\]](#)
3. Chmielewski, M.; Kukielka, M.; Frąszczak, D.; Bugajewski, D. Military and Crisis Management Decision Support Tools for Situation Awareness Development Using Sensor Data Fusion. In *Information Systems Architecture and Technology: Proceedings of 38th International Conference on Information Systems Architecture and Technology—ISAT 2017*; Świątek, J., Borzemski, L., Wilimowska, Z., Eds.; Springer: Cham, Switzerland, 2018; pp. 189–199.
4. Pasdar, A.; Koroniotis, N.; Keshk, M.; Moustafa, N.; Tari, Z. Cybersecurity Solutions and Techniques for Internet of Things Integration in Combat Systems. *IEEE Trans. Sustain. Comput.* **2025**, *10*, 345–365. [\[CrossRef\]](#)
5. Bout, E.; Loscri, V.; Gallais, A. How Machine Learning Changes the Nature of Cyberattacks on IoT Networks: A Survey. *IEEE Commun. Surv. Tutor.* **2022**, *24*, 248–279. [\[CrossRef\]](#)
6. Ambrosin, M.; Anzanpour, A.; Conti, M.; Dargahi, T.; Moosavi, S.R.; Rahmani, A.M.; Liljeberg, P. On the feasibility of attribute-based encryption on internet of things devices. *IEEE Micro* **2016**, *36*, 25–35. [\[CrossRef\]](#)
7. Pióro, L.; Sychowiec, J.; Kanciak, K.; Zieliński, Z. Application of Attribute-Based Encryption in Military Internet of Things Environment. *Sensors* **2024**, *24*, 5863. [\[CrossRef\]](#) [\[PubMed\]](#)
8. Szczegielniak-Rekiel, A.; Kanciak, K.; Kelner, J.M. Zero-Knowledge Proof in 5G and Beyond Technologies: State of the Arts, Practical Aspects, Applications, Security Issues, Open Challenges, and Future Trends. *IEEE Access* **2025**, *13*, 138352–138380. [\[CrossRef\]](#)
9. Zaidi, S.Y.A.; Shah, M.A.; Khattak, H.A.; Maple, C.; Rauf, H.T.; El-Sherbeeney, A.M.; El-Meligy, M.A. An Attribute-Based Access Control for IoT Using Blockchain and Smart Contracts. *Sustainability* **2021**, *13*, 10556. [\[CrossRef\]](#)
10. Rouhani, S.; Belchior, R.; Cruz, R.S.; Deters, R. Distributed attribute-based access control system using permissioned blockchain. *World Wide Web* **2021**, *24*, 1617–1644. [\[CrossRef\]](#)

11. Bhatt, S.; Pham, T.K.; Gupta, M.; Benson, J.; Park, J.; Sandhu, R. Attribute-based access control for AWS internet of things and secure industries of the future. *IEEE Access* **2021**, *9*, 107200–107223. [\[CrossRef\]](#)
12. Bethencourt, J.; Sahai, A.; Waters, B. Ciphertext-policy attribute-based encryption. In Proceedings of the 2007 IEEE Symposium on Security and Privacy (SP'07), Berkeley, CA, USA, 20–23 May 2007; IEEE: Piscataway, NJ, USA, 2007; pp. 321–334.
13. Johnsen, F.T.; Zieliński, Z.; Wrona, K.; Suri, N.; Fuchs, C.; Pradhan, M.; Furtak, J.; Vasilache, B.; Pellegrini, V.; Dyk, M.; et al. Application of IoT in military operations in a smart city. In Proceedings of the 2018 International Conference on Military Communications and Information Systems (ICMCIS), Warsaw, Poland, 22–23 May 2018; IEEE: Piscataway, NJ, USA, 2018; pp. 1–8.
14. Cha, S.; Baek, S.; Kang, S.; Kim, S. Security evaluation framework for military IoT devices. *Secur. Commun. Netw.* **2018**, *2018*, 6135845. [\[CrossRef\]](#)
15. Hu, V.C.; Ferraiolo, D.F.; Chandramouli, R.; Kuhn, D.R. *Attribute-Based Access Control*; Artech House: Norwood, MA, USA, 2017.
16. Celesti, A.; Fazio, M.; Giacobbe, M.; Puliafito, A.; Villari, M. Characterizing cloud federation in IoT. In Proceedings of the 2016 30th International Conference on Advanced Information Networking and Applications Workshops (WAINA), Crans-Montana, Switzerland, 23–25 March 2016; IEEE: Piscataway, NJ, USA, 2016; pp. 93–98.
17. Pradhan, M.; Noll, J. Security, privacy, and dependability evaluation in verification and validation life cycles for military IoT systems. *IEEE Commun. Mag.* **2020**, *58*, 14–20. [\[CrossRef\]](#)
18. Arora, A.S.; Raja, L.; Bahl, B. Data centric security approach: A way to achieve security & privacy in cloud computing. In Proceedings of the 3rd International Conference on Internet of Things and Connected Technologies (ICIOTCT), Jaipur, India, 26–27 March 2018; pp. 26–27.
19. Wrona, K. Securing the Internet of Things: A Military Perspective. In Proceedings of the 2015 IEEE 2nd World Forum on Internet of Things (WF-IoT), Milan, Italy, 14–16 December 2015; pp. 502–507. [\[CrossRef\]](#)
20. Brézot, T.; de Perthuis, P.; Pointcheval, D. Covercrypt: An Efficient Early-Abort KEM for Hidden Access Policies with Traceability from the DDH and LWE. In *European Symposium on Research in Computer Security*; Springer: Cham, Switzerland, 2023.
21. Khandla, D.; Shahy, H.; Bz, M.K.; Pais, A.R.; Raj, N. Expressive CP-ABE Scheme Satisfying Constant-Size Keys and Ciphertexts. *Cryptol. ePrint Arch.* **2019**, 1257.
22. Cosmian. Cosmian/Cover_CRYPT: Implementation of the Covercrypt Attribute-Based KEMAC. 2025. Available online: https://github.com/Cosmian/cover_crypt (accessed on 16 July 2025).
23. de la Piedra, A.; Venema, M.; Alpár, G. ABE Squared: Accurately Benchmarking Efficiency of Attribute-Based Encryption. *IACR Trans. Cryptogr. Hardw. Embed. Syst.* **2022**, *2022*, 192–239. [\[CrossRef\]](#)
24. Venema, M. Attaining Basically Everything in Attribute-Based Encryption. Ph.D. Thesis, University of Groningen, Groningen, The Netherlands, 2023.
25. Bayuk, J. Data-centric security. *Comput. Fraud. Secur.* **2009**, *2009*, 7–11. [\[CrossRef\]](#)
26. Wrona, K. Towards Data-Centric Security for NATO Operations. In *Proceedings of the Digital Transformation, Cyber Security and Resilience*; Tagarev, T., Stoianov, N., Eds.; Springer: Cham, Switzerland, 2024; pp. 75–92.
27. Fagan, M.; Marron, J.; Souppaya, M.; Watrobski, P.; Kent, K.; Mulugeta, B.; Symington, S. *Towards Automating IoT Security: Implementing Trusted Network-Layer Onboarding*; NIST Cybersecurity White Papers (CSWP); National Institute of Standards and Technology: Gaithersburg, MD, USA, 2025.
28. Mosteiro-Sanchez, A.; Barcelo, M.; Astorga, J.; Urbiet, A. Too Many Options: A Survey of ABE Libraries for Developers. *arXiv* **2022**, arXiv:2209.12742. [\[CrossRef\]](#)
29. National Institute of Standards and Technology (NIST). *Guide to Attribute Based Access Control (ABAC) Definition and Considerations*; NIST SP Special Publication 800-162; NIST: Gaithersburg, MD, USA, 2014.
30. Bernstein, D.J.; Lange, T. Post-quantum cryptography. *Nature* **2017**, *549*, 188–194. [\[CrossRef\]](#) [\[PubMed\]](#)
31. Alagic, G.; Apon, D.; Cooper, D.; Dang, Q.; Dang, T.; Kelsey, J.; Lichtinger, J.; Liu, Y.K.; Miller, C.; Moody, D.; et al. *Status Report on the Third Round of the NIST Post-Quantum Cryptography Standardization Process*; NIST Internal Report NISTIR 8413; NIST: Gaithersburg, MD, USA, 2022.
32. European Telecommunications Standards Institute (ETSI). *Efficient Quantum-Safe Hybrid Key Exchanges with Hidden Access Policies*; ETSI: Sophia Antipolis, France, 2025.
33. National Institute of Standards and Technology. *FIPS 203: Module-Lattice-Based Key-Encapsulation Mechanism Standard*; Technical Report; NIST: Gaithersburg, MD, USA, 2024. [\[CrossRef\]](#)
34. Brézot, T.; Héban, C.; de Perthuis, P.; Pointcheval, D. Security Analysis of Covercrypt: A Quantum-Safe Hybrid Key Encapsulation Mechanism for Hidden Access Policies. In *European Symposium on Research in Computer Security*; Springer: Cham, Switzerland, 2025.
35. Tomida, J.; Kawahara, Y.; Nishimaki, R. Fast, Compact, and Expressive Attribute-Based Encryption. *Cryptol. ePrint Arch.* **2019**, 966.
36. CIRCL: Cloudflare Interoperable, Reusable Cryptographic Library. Implements CP-ABE on BLS12-381; Optimized Pairing Routines. 2019. Available online: <https://github.com/cloudflare/circl> (accessed on 4 November 2025).

37. Wang, X.; Zhang, J.; Schooler, E.M.; Ion, M. Performance evaluation of Attribute-Based Encryption: Toward data privacy in the IoT. In Proceedings of the 2014 IEEE International Conference on Communications (ICC), Sydney, NSW, Australia, 10–14 June 2014; pp. 725–730. [[CrossRef](#)]
38. Perazzo, P.; Righetti, F.; La Manna, M.; Vallati, C. Performance evaluation of Attribute-Based Encryption on constrained IoT devices. *Comput. Commun.* **2021**, *170*, 151–163. [[CrossRef](#)]
39. Shruti; Rani, S.; Sah, D.K.; Gianini, G. Attribute-Based Encryption Schemes for Next Generation Wireless IoT Networks: A Comprehensive Survey. *Sensors* **2023**, *23*, 5921. [[CrossRef](#)] [[PubMed](#)]
40. Lewko, A.; Waters, B. Decentralizing Attribute-Based Encryption. In Proceedings of the Advances in Cryptology–EUROCRYPT 2011, Tallinn, Estonia, 15–19 May 2011; Paterson, K.G., Ed.; Springer: Berlin/Heidelberg, Germany, 2011; pp. 568–588.

Disclaimer/Publisher’s Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.