applied sciences

MDPI

Article

Token-Revocation Access Control to Cloud-Hosted Energy Optimization Utility for Environmental Sustainability

Khaled Riad [1,2]

[1] Computer Science Department, College of Computer Sciences & Information Technology, King Faisal University, Al-Ahsa 31982, Saudi Arabia; kriad@kfu.edu.sa or khaled.riad@science.zu.edu.eg
[2] Mathematics Department, Faculty of Science, Zagazig University, Zagazig 44519, Egypt

Abstract: To increase the usage of renewable energy, it is vital to maximize local energy production by properly combining various renewable-energy sources by collecting their data and storing it on the cloud. The energy optimization utility, which is used for making decisions to optimize renewable-energy resources, is hosted on the cloud to benefit from cloud capabilities in data storage. Hosting such sensitive data and utilities on the cloud has created some cybersecurity challenges. This paper presents a new token-revocation access control (TR-AC) which revokes the authorization of malicious users before authorizing them to access cloud-hosted energy optimization utilities. TR-AC employs a set of multi-authorities to measure the authentic level for each authenticated user. Although the user is authenticated to access the online system, this authentication can be revoked to utilize the energy optimization utility based on the user's level of authentication. The cloud storage servers are not fully trusted and, therefore, have no control over access controls. Finally, the proposed TR-AC has been proven to be secure against any attacker that is not authentic according to Diffie-Hellman assumptions. In addition, performance analysis has proven that the time elapsed for both encryption and decryption in TR-AC is very small compared with previously introduced schemes. Therefore, it will not affect the performance of the cloud-hosted system.

Keywords: access control; energy optimization utility; environmental sustainability; cloud security; user revocation

Citation: Riad, K. Token-Revocation Access Control to Cloud-Hosted Energy Optimization Utility for Environmental Sustainability. *Appl. Sci.* **2023**, *13*, 3142. https://doi.org/10.3390/app13053142

Academic Editors: Tuan Anh Nguyen and Francisco Airton Silva

Received: 9 January 2023
Revised: 23 February 2023
Accepted: 24 February 2023
Published: 28 February 2023

1. Introduction

It is uncommon to pair cybersecurity with sustainability in the same sentence. However, in order to achieve environmental sustainability, high-level cybersecurity features must be integrated into decision-making energy optimization tools and utilities [1]. Hosting the data collected from renewable-energy sources and energy optimization utility on the cloud has created some cybersecurity challenges. Thus, decision-makers and providers must plan and mitigate cybersecurity threats. In light of this, the renewable-energy sector's quick evolution has exposed some major cybersecurity challenges [2].

The first challenge is related to employing Internet-of-Things (IoT) devices to collect data from renewable-energy sources. In addition, IoT is used in controlling energy loads, promoting industrial operational effectiveness, and offering a more flexible energy experience [3]. On the other hand, low-cost IoT devices have many security vulnerabilities. These may allow attacks on energy systems through backdoor gateways [4]. Due to their apparent passivity, seemingly unprotected devices such as a thermostat, printer, or industrial sensor can be used to climb the technology stack and gain access to more vital networks [5]. To the best of our knowledge, many energy operators lack the necessary visibility to effectively defend these sensitive networks.

The second challenge represents the increased number of attacks on the renewable-energy sector due to the development of more connected industrial assets and large-scale operating technologies [6]. The third challenge is related to storing the collected renewable-energy data on the cloud, which is a decentralized environment for easy and remote access.

This creates the need for efficient and secure identity verification and access management. This access-management scheme should work as an efficient protection layer between the user requesting access and the sensitive renewable-energy data hosted in the cloud [7].

The first step in effective access control is authentication. While authorization is made possible via authentication, which confirms the user's identity, after the authentication process, a user is given the right to perform something, which is referred to as authorization [8]. For example, a user requesting access can create and use an identity to log into the energy optimization utility website, but the optimization utility policy must ensure that. After verifying the user's identity, they are authorized to monitor some fields from the hosted database, such as solar, wind, hydro, geothermal, and biomass energy. However, this authenticated user cannot specify constraints in the model to investigate the optimal renewable-energy combination. Thus, the optimization utility authorization policy can be used at more specific levels than for access alone.

A crucial step in preventing unauthorized access to information and the exploitation of computer systems is the proper setting of access privileges [9]. Therefore, to protect sensitive renewable-energy data and the energy optimization utility, a secure, flexible, and privacy-preserving access control system must exist. Renewable-energy organizations can employ solutions which include encryption, two-factor authentication, and authorization built in as a security feature. This paper introduces the evidence for using token-revocation access control as a feasible solution for the renewable-energy sector's security against several high-risk cyber attacks, such as malware infection and data leakage. Successful experiments with different renewable-energy organizations demonstrated that the rigorous implementation of token-revocation access control can effectively mitigate a wide range of cyber threats without introducing a lot of additional complexity or cost. The token-revocation access-control scheme has the ability to revoke the user's access at various points based on a constantly shifting revocation threshold.

1.1. Motivation

The central authority in charge of access control may serve as a primary target for assault. Additionally, centralized control is widely favored in order to protect and guarantee the reliability of renewable-energy operations. However, adaptability and system integration at each organization is equally crucial. In order to secure the greatest return on investment and simplicity of installation, the access-control system should also provide interoperability with other third-party and legacy systems. To the best of our knowledge, the current access-control schemes suffer from the following main points:

- Categorizing the users into a limited number of roles that can be used to access the encrypted data, which leads to the role explosion problem.
- Allowing the individual authorities to decrypt the ciphertexts, which violates data privacy.
- Placing high importance on the central authority in managing huge computations, which makes it a bottleneck and vulnerable to attack.
- Revoking the users only based on the attributes of the user. However, the revocation should be based on the user's behavior.

We were inspired by the aforementioned points to suggest a token-revocation access-control system for the cloud-hosted renewable-energy data and energy optimization utility. TR-AC has the ability to block malicious users at the authorization stage using the authentic token algorithm.

1.2. Main Contributions

Due to the sensitivity of the renewable-energy data and the negatives consequences of the energy optimization utility being accessed by a malicious user, this sector must not depend on only one authority for approving access to cloud-hosted resources. In other words, it is better to distribute the access decision among multiple authorities. This paper generates an effective token to validate the access request for a specific user using

multiple authorities distributed in various geographical locations. The proposed token was implemented in a complete token-revocation access-control system. The main contributions are as follows:

- The proposed scheme has a pool of dynamically changing revocation thresholds. Then, If the malicious user was able to pass through some stages of the access-control scheme, it will definitely be revoked in one of the later stages.
- The proposed system was proven to be secure against any attacker, based on the Diffie–Hellman assumption. In addition, the time elapsed in evaluating each user until either granting or denying access is very short in comparison with previously introduced access-control systems.
- The encryption and decryption times for the proposed system were compared with LW [10], LCHWY [11], MAACS [12], and MD-AC [13].

2. Related Literature

The traditional access-control systems [14–17] are the main access-control schemes. The Discretionary Access Control (DAC) model [14] allows object owners to limit access to their objects or the information contained in them depending on the identities of users or membership of particular groups. The mandatory access control (MAC) model [15] protects against direct and indirect information leaks, but perfect information privacy cannot be ensured. As the DAC model is typically less safe than the MAC model, it is utilized in settings where a high level of security is not necessary [18]. The idea underlying the role-based access control (RBAC) model [19] is that "a subject's duty is more essential than the subject's identity." In the attribute-based access control (ABAC) model [17], the attributes can be defined or used in a variety of ways. However, in cloud computing, it might be difficult to reach a consensus on what kind of qualities should be used and how many traits should be used when making access decisions [20].

The authors of [21] introduced CoRBAC. It is considered an RBAC system. It focuses on introducing access control to cloud computing. The domain model and role model of distributed RBAC are carried over. It can combine the distributed authentication services provided by the CoRBAC, in order to give the *BiC* the ability to issue certificates. CoRBAC has some significant drawbacks. The authors in [22] introduced adaptive extensible access control markup language (XACML) access policies or heterogeneous distributed Internet of Things (IoT) environments. The authors in [23] employed blockchain technology to introduce a privacy-preserving access-control scheme for securely sharing electronic health records. Another notarization-based authorization model was proposed by Chakraborty and Ray [24].

Allison Lweko and Brent Waters introduced the LW scheme [10] which decentralizes the attribute-based encryption. This scheme still depends on categorizing users into the limited number of roles that can be used to access the encrypted data. Zhen Liu et al. [11] introduced the LCHWY scheme. In this scheme, the individual authorities are able to decrypt ciphertexts. The authors in [12] introduced the MAACS scheme. In this scheme, the decryption overhead at the client side has been eliminated but the revocation process in MAACS is only based on the attributes of the user. It does not consider revoking the user based on his behavior. The authors in [13] introduced the MD-AC scheme. This scheme places importance on the central authority in managing huge computations which makes it a single point of bottleneck and attack.

3. TR-AC: Token-Revocation Access Control

Our TR-AC scheme computes an authentic token for each user requesting access. Thus, only authentic users will be able to access the cloud-hosted renewable-energy data and energy optimization utility. The encryption algorithm defines the privileges that a user grants [25]. Moreover, the authorized permission is used with the calculated authenticity level to be consistent with the revocation threshold. The proposed TR-AC has four main algorithms.

The main model framework shown in Figure 1 is composed of the following entities:

- **Organizations' sensitive data and utilities:** This is the renewable-energy data and energy optimization utilities. It will be encrypted. Then, it will be hosted on the cloud storage. In addition, efficient access control will be used to manage access to energy optimization utilities.
- **Users:** There is a huge number of users who may request access to the cloud-hosted renewable-energy data and energy optimization utility, through a variety of different devices. A User ID (UID) is given to each user who requests access to renewable-energy organization cloud data and energy optimization utilities. This UID is given by the renewable-energy organization and never changes for that user.
- **Organization Policy.** It is the decision-making policy for each organization's access control. It is identified by a unique ID (OID).
- **Authentication servers.** The user identified by UID enters his user name, password, and any additional information necessary for successful authentication in this stage. The authentication servers have a set of servers under their own management. Each of these servers has a specified task.
- **Multi-authority system.** It is a collection of various authority corresponding with one another. By determining the *users' authentication level*, they are in charge of assessing the user before granting him any access. The user is granted temporary access to the cloud renewable-energy data if the prganization in charge (OiC) and other system authorities trust their decision to grant or deny access. In this section, a secret key (SK_{UID}) is given to each user identified by a unique UID.
- **Cloud Storage.** A set of servers with enormous capacities to manage the voluminous demands for data storage and access.

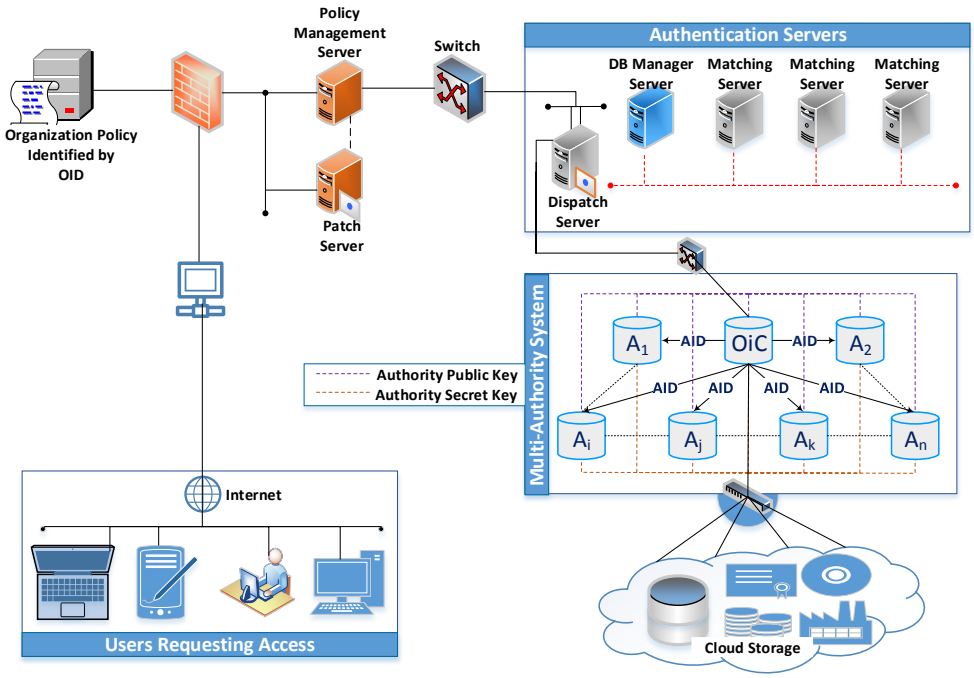


Figure 1. The proposed TR-AC scheme integrated with the renewable-energy organization authentication system.

The detailed flowchart for a user identified by UID requesting access to the cloud-hosted resources is illustrated in Figure 2. Firstly, the user requests access and provides his/her credentials. Then, the authentication servers with the help of the organization's

policy will evaluate the provided credentials to decide whether the user is authenticated or not. If the user is not authenticated, s/he cannot go further in the model. If the user is authenticated, then the user's attributes will be provided to the multi-authority system to calculate the weighted authentication token. After that, the authentic token generation (Algorithm 1) runs to decide whether the user is revoked or not. If the user is revoked, s/he receives a random hexadecimal decryption token and cannot go further in the model. If the user was non-revoked, s/he will receive a valid decryption token. This decryption token will be used in the decryption (Algorithm 2). Finally, the non-revoked user will obtain the original data or requested service.

Algorithm 1 Authentic token generation

Input: - Access structure $\mathcal{AC}$
- UID attributes AT_{UID}
- UID Weighted authentic $WAL_{UID} = [wb_{UID}, wd_{UID}, wu_{UID}]t$
- Initial threshold vector $\tilde{N}_t = [b_{\tilde{N}}, d_{\tilde{N}}, u_{\tilde{N}}]_t$

Consider $Authentic = 0$
if $(wb_{UID} > b_{\tilde{N}})$ **then**
  $Authentic = 1$ ▷ UID is authentic
else if $(wb_{UID} = b_{\tilde{N}} \wedge wd_{UID} < d_{\tilde{N}})$ **then**
  $Authentic = 1$ ▷ UID is authentic
else if $(wb_{UID} = b_{\tilde{N}} \wedge wd_{UID} = d_{\tilde{N}} \wedge wu_{UID} < u_{\tilde{N}})$ **then**
  $Authentic = 1$ ▷ UID is authentic
else
  $Authentic = 0$ ▷ UID is not authentic
end if
Select $\delta_i \in \mathbb{Z}_p^*$
for $i = 1$ to $|I|$ **do**
  if $\lambda_i \in Share(f)$ **then**
    Reconstruct $f = \sum_{i \in I} \delta_i \lambda_i$
  end if
end for
if $AT_{UID} \vDash \mathcal{AC} \wedge Authentic$ **then**

$$DT = \prod_{k=1}^{N_A} e(g,g)^{\frac{\alpha_k}{z_j} f}$$

else
  $DT = rand(Hex)$
end if

Output: The authentic Decryption Token DT

Algorithm 2 Decryption

Input: - The ciphertext $CT = \{CT_k\}$
- One component from the CT_k $\{C_k = ck_k \cdot (\prod_{k \in K} e(g,g)^{\alpha_k})^f\}$
- Authentic decryption token $DT = \prod_{k=1}^{N_A} e(g,g)^{\frac{\alpha_k}{z_j} f}$
- UID secret key $SK_{c_j} = z_j$

for $k = 1$ to N **do**
  $ck_k = \frac{C}{DT^{z_j}}$
  Calculate $\mathcal{D}_k$ from CT_k using ck_k
end for
Find $\mathcal{D}$ using $\{\mathcal{D}_k\}$

Output: Original data $\mathcal{D}$

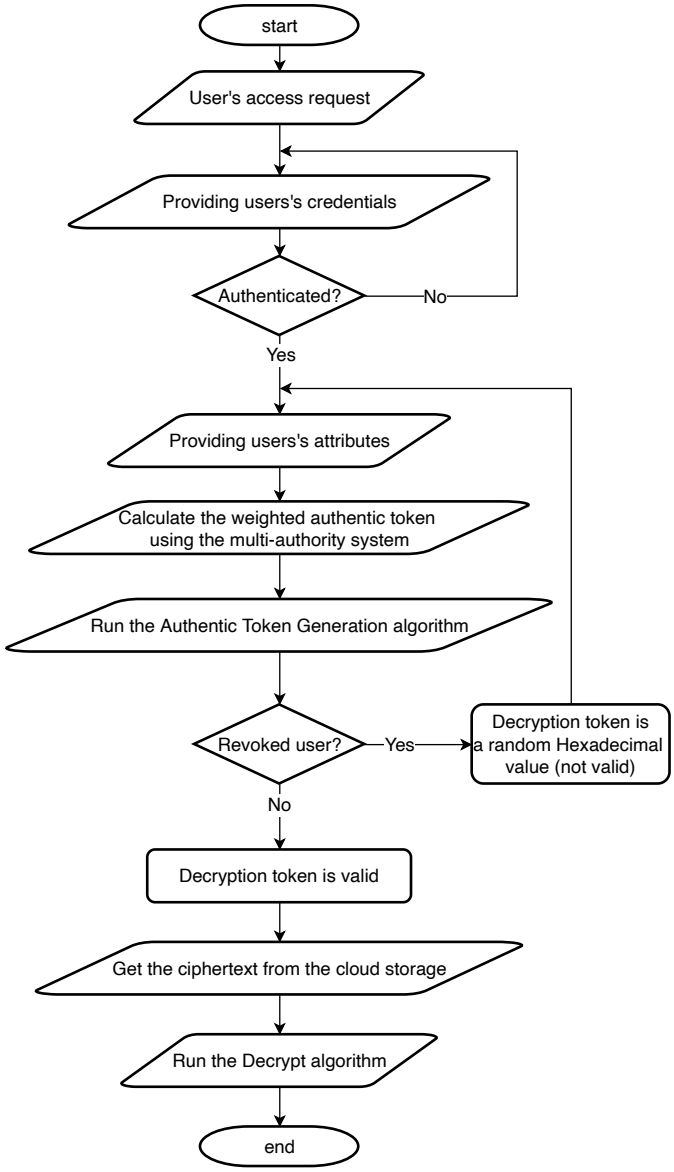


Figure 2. The detailed flow diagram for a user requesting access to the cloud-hosted services and data.

3.1. Setup

Let the bilinear group $\mathcal{B} = (p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e(\cdot,\cdot))$, where ($\mathbb{G}_1 = \mathbb{G}_2 = \mathbb{G}$). Consider $g \in \mathbb{G}$. Randomly compute $\{\alpha_k, \beta_k, and\ \gamma_k\}_{\forall k \in K} \in \mathbb{Z}_p^*$. Let $MK = (\beta_k, g^{\alpha_k})$. Consider $H : \{0,1\}^* \rightarrow \mathbb{G}_1$ as a hash function. It has four sub-algorithms:

- **OiC-Setup.** It initializes *OiC*. In addition, it shares S_0 among the system authorities. $w_{k,i} \in [0,1]$ is selected for each k in K and $i \in [1,n]$. $(a_{m,k} = a_{k,i})$ is an attribute $\forall AID$. Each attribute has a weight ($wa_{m,k} = \lfloor w_{k,i} \cdot 11^s \rfloor$, and $s \in \mathbb{Z}_q^*$). A parameter ($SPK = e(g,g)^{S_0}$) is initialized here. An initial threshold is $\tilde{N}_t = [b_{\tilde{N}}, d_{\tilde{N}}, u_{\tilde{N}}]_t$ at time t.
- **UID-Setup.** The user is assigned a unique ID (UID), and $PK_{UID} = g^{s_{UID}}$. The secret key is $SK_{UID} = v_{UID}$ with $s_{UID}, v_{UID} \in \mathbb{Z}_p^*$.
- **AID-Setup.** Each authority is assigned a unique Authority ID (AID) by *OiC*. It generates two keys:
  - **Attribute-secret key ($\{aSK_{a_i}^k\}_{\forall a_i \in At_k}$):** $\{aSK_{a_i}^k\}_{\forall a_i \in At_k} = \{(\alpha_k, \beta_k, \gamma_k)\}_{\forall k \in K}$. It will be used by the organizations.

– **Attribute-public key ($\{aPK_{a_i}^k\}_{\forall a_i \in At_k}$):** $\{aPK_{a_i}^k\}_{\forall a_i \in At_k}$. It will be used by the users. $\{aPK_{a_i}^k\}_{\forall a_i \in At_k} = (g^{v_{a_{m,k}}} \cdot H(a_{m,k}))^{\gamma_k}$ by implicitly choosing $v_{a_{m,k}} = g^{w_{a_{m,k}}}$.

3.2. Key Generation

Each authority identified by a unique AID generates two keys ($aSK_{a_i \forall} a_i \in At_k$ and $aPK_{a_i \forall} a_i \in At_k$). Each AID chooses $\{\alpha_k, \beta_k, \gamma_k\}_{\forall k \in K} \in \mathbb{Z}_p^*$ randomly.

• **Gen-PK.** Each authority generates a public key as follows:

$$\{PK_k = (e(g,g)^{\alpha_k}, g^{\frac{1}{\beta_k}}, g^{\frac{\gamma_k}{\beta_k}})\}_{\forall k \in K}.$$

• **Gen-SK.** Each manging authority AID generates a secret key $(SK_{c_j}^k)$ for each user UID after verifying the UID provided attributes, as follows:

$$\{K_{c_j,k} = g^{\frac{\alpha_k}{z_j}} \cdot g^{a \cdot c_j} \cdot g^{\frac{a}{\beta_k} t_{j,k}}\}_{\forall j; \forall k},$$

$$\{R_{c_j,k} = g^{a \cdot t_{j,k}}\}_{\forall j; \forall k}, \text{ and } \{L_{c_j,k} = g^{\frac{\beta_k}{z_j} t_{j,k}}\}.$$

$\forall \{a_{j,k} \in At_j^k\}$ with value $v_{a_{j,k}} \in \mathbb{Z}_p^*$,
compute:

$$\{K_{c_j,a_{j,k}} = g^{\frac{\beta_k \gamma_k}{z_j} t_{j,k}} \cdot (g^{v_{a_{j,k}}} \cdot H(a_{j,k}))^{\gamma_k \beta_k c_j}\}_{\forall j; \forall k}$$

The secret key is:

$$\{SK_{c_j}^k = (K_{j,k}, L_{j,k}, R_{j,k}, K_{j,a_{j,k}})\}_{\forall j; \forall k}$$

3.3. Encryption

The used encryption access structure is $\mathcal{AC} = (M, \rho)$. The encryption algorithm is introduced in Algorithm 3.

Algorithm 3 Encryption

Input: - Content key ck_k for each AID
- Public key PK_k for each AID
- Access structure $\mathcal{AC}$)

Select exponents $\{\alpha_k, \beta_k, \gamma_k\}_{\forall k \in K}$
Select a valid encryption exponent f
Select a vector $\vec{v} = [f, y_2, \cdots, y_n]^T$
for $i = 1$ to l **do**
    Find $\lambda_i = (M\vec{v})_i$
end for
Select $r_1, \cdots, r_l \in \mathbb{Z}_p^*$
Find $\{C' = g^{\frac{f}{\beta_k}}\}$
for $i = 1$ to l **do**
    Find $\{C_i = g^{a\lambda_i} \cdot ((g^{\vec{v}(i)\rho(i)} \cdot H(\rho(i))^{\gamma_k}))^{-r_i}\}_{\forall k \in K}$
    Compute $\{D_{1,i} = g^{\frac{r_i}{\beta_k}}\}$ and $\{D_{2,i} = g^{-\frac{\gamma_k}{\beta_k} r_i}\}$
end for
Compute the ciphertext:

$$\{CT\}_{\forall k \in K} = [C = ck_k \cdot (\prod_{k \in K} e(g,g)^{\alpha_k})^f, C', C_i, D_{1,i}, D_{2,i}, \rho(i)]_{i=1:l}$$

Output: The ciphertext CT

3.4. Decryption

The details of the decryption stage are as follows:

- **Authentic token generation.** The decryption token is computed at the server side, by executing the Token-Generation algorithm. In TR-AC, if the UID weighted authentic level $WAL_{UID} = [wb_{UID}, wd_{UID}, wu_{UID}]_t$ overcome $\tilde{N}_t$, based on the comparison between two authentic levels defined in [13], the authentic decryption token generated by UID.

  The detailed authentic token generation process is shown in Algorithm 1. The UID has a secret key $SK_{c_j} = z_j$ and the managing AID will provide the public key $PK_{c_j} = g^{c_j}$. Then, $c_j, z_j \in \mathbb{Z}_p^*$ is randomly chosen. In addition, $s_{c_j} \in \mathbb{Z}_p^*$, and $\{\delta_i\}_{\forall i \in I}$ are randomly chosen. The exponent $f = \sum_{i \in I} \delta_i \lambda_i \,|\, f \in \mathbb{Z}_p^*$ is reconstructed. Where $\{\lambda_i\}_{\forall i \in I}$ are valid, f shares:

$$\begin{aligned} DT &= \prod_{k=1}^{N_A} \frac{e(C', K_{c_j,k}) \cdot e(R_{c_j,k}, C'')^{-1}}{\prod_{i \in I} [e(C_i, GPK_{c_j}) \cdot e(D_{1,i}, K_{c_j,\rho(i)}) \cdot e(D_{2,i}, L_{c_j,k})]^{\delta_i}} \\ &= \frac{e(g,g)^{f \cdot a \cdot c_j \cdot N_A} \prod_{k=1}^{N_A} e(g,g)^{\frac{f \cdot \alpha_k}{z_j}}}{e(g,g)^{a \cdot c_j \cdot N_A \cdot f}} \\ &= \prod_{k=1}^{N_A} e(g,g)^{\frac{\alpha_k}{z_j} f} \end{aligned} \tag{1}$$

  If the user was unable to successfully generate the authentic decryption token, he will not be able to generate the original data $\mathcal{D}$.

- **Decryption.** After generating the $(DT = \prod_{k=1}^{N_A} e(g,g)^{\frac{\alpha_k}{z_j} f})$ using the UID with c_j using Algorithm 1. The user can obtain the original data by decrypting the ciphertext using a decryption token and his secret key $SK_{c_j} = z_j$. The content key will be generated as follows:

$$\{ck_k = \frac{C}{DT^{z_j}}\}_{\forall k=1:N_A}$$

  where $C = ck_k \cdot (\prod_{k=1}^{N_A} e(g,g)^{\alpha_k})^f$. The generated ck_k is used to decrypt the ciphertext.

  Algorithm 2 is the detailed decryption algorithm. It takes as input the ciphertext to be decrypted, $(CT = \{CT_k\}_{\forall k \in K})$. It has multiple components The first component $\{C_k = ck_k \cdot (\prod_{k \in K} e(g,g)^{\alpha_k})^f\}_{\forall k \in K}$ generates the content key for that authority by the user. $(DT = \prod_{k=1}^{N_A} e(g,g)^{\frac{\alpha_k}{z_j} f})$, and $(GSK_{c_j} = z_j)$. It outputs the decrypted data $(\mathcal{D})$ by recovering it from $\{CT_k\}_{\forall k \in K}$ using ck_k for each AID.

3.5. Implementation

The introduced scheme and its algorithms were implemented on top of our private cloud environment built using OpenStack (https://www.openstack.org, accessed: 6 January 2023). Our private cloud environment is based on three physical servers (controller, network, and nova-compute). The configuration of the controller node and network node is 48 core CPUs, 128 GB RAM, and a 5 TB disk. The configuration of the nova-compute node is a 24 core CPUs, 128 GB RAM, and a 2 TB disk. The introduced access control system was implemented using the following main structures of virtual machines (VMs):

- **Tiny VMs:** It is composed of 300 VMs. The configuration of each VM is 1 virtual CPU, 512 MB RAM, and 1 GB disk. These VMs are used as the users requesting access to the introduced access-control system.

- **Small VMs:** It is composed of 32 VMs. The configuration of each VM is 1 virtual CPU, 2048 MB RAM, and a 20 GB disk. These VMs are used as the system authorities identified by AID.
- **Medium VMs:** It is composed of five VMs. The configuration of each VM is 2 virtual CPUs, 4096 MB RAM, and a 40 GB disk. These VMs are used for the organization's policy (OID), policy management server, patch server, and the organization central authority (OiC).
- **Large VMs:** It is composed of five VMs. The configuration of each VM is four virtual CPUs, 8192 MB RAM, and 80 GB disk. These VMs are used for the authentication servers.

4. Security Analysis

The proposed TR-AC scheme is proved secure based on the following theorems.

Theorem 1. *Neither the cloud servers nor the unauthorized users can compromise the components of the ciphertext and the authority content key based on the Diffe–Hellman (DH) assumption.*

Proof. In the encryption stage, which is executed using Algorithm 3, this algorithm is executed by the cloud servers. The input is a set of components that are useful in generating a secure ciphertext, such as the access structure $\mathcal{AC}$ which is provided by the system authorities. These inputs are secure and cannot be recovered by the cloud server. The most important one is the authority content key because it will be recovered first in the decryption Algorithm 2, in order to recover the original data. After that, the encryption algorithm computes the variables that compromise the ciphertext:

- $C = ck_k \cdot (\prod_{k \in K} \hat{e}(g,g)^{\alpha_k})^f$, which is computed based on the content key ck_k, so it is secure and cannot be recovered by unauthorized users;
- $C' = g^f$, where $f \in \mathbb{Z}_q^*$ is the random encryption exponent and g^f, which is secure based on the Diffie–Hellman (DH) assumption. Given g and g^f, it is hard to find f, because it is a discrete logarithm (DL) problem. Thus, the value of C' cannot be computed by an attacker;
- $\{C'' = g^{\frac{f}{\beta_k}}\}_{\forall k \in K}$, which is also secure based on the Diffie–Hellman (DH) assumption;
- $\{C_i = g^{a\lambda_i} \cdot ((g^{\vec{v}(i)\rho(i)} \cdot H(\rho(i))^{\gamma_k}))^{-r_i}\}_{\forall k \in K}$. According to the first component $(g^{a\lambda_i})$, given as g and $g^{a\lambda_i}$, it is hard to find λ_i for two reasons, because it is a DL problem and because it is generated using linear secret sharing (LSSS) as follows: To generate the shares for a secret $f \in \mathbb{Z}_q^*$, we choose a column vector $\overrightarrow{v} = (s, r_2, \cdots, r_n)^T$ where $r_2, \cdots, r_n$ are randomly picked from $\mathbb{Z}_q^*$, then $M\overrightarrow{v}$ is the vector of ℓ shares of f according to the sharing $\sum$, and the share $(M\overrightarrow{v})_i$ belongs to the party $\rho(i)$.
  Accordingly, the second part $((g^{\vec{v}(i)\rho(i)} \cdot H(\rho(i))^{\gamma_k}))$ is based on the calculation of the hash function $H(\rho(i))$. Let us consider $g^r = H(\rho(i)) \in \langle g \rangle$, where g is the generator of $\mathbb{G}$ and $\langle g \rangle = g \cdot g^2 \cdot g^3 \cdots g^{q-1} \cdot g^q$ where $g^q = g$. Thus, given g and g^r it is hard to find r, because it is a DL problem. In addition, the hash function $H(\cdot)$ is a one-way function, and $\gamma_k \in \mathbb{Z}_q^*$ is a random constant. Thus, the attacker cannot compute the value of C_i and
- $\{D_{1,i} = g^{\frac{r_i}{\beta_k}}\}_{\forall k \in K}$ and $\{D_{2,i} = g^{-\frac{\gamma_k}{\beta_k} r_i}\}_{\forall k \in K}$ are also computed based on the values of some random constants such as $\beta_k, \gamma_k \in \mathbb{Z}_q^*$ and are secure based on the DH assumption, because it is DL problem. Therefore, an unauthorized user cannot compute the values of $D_{1,i}$ and $D_{2,i}$.

Therefore, the encryption algorithm is secure and each component in the ciphertext is secure and cannot be computed by unauthorized sets of users. □

Theorem 2. *The user can successfully generate the authentic token and decrypt the ciphertext only if it has a set of attributes that are a part of the $\mathcal{AC}$ for the requested ciphertext.*

Proof. The decryption token will be successfully generated based on the following corrected proof for generating DT:

$$DT = \prod_{k=1}^{N_A} \frac{\hat{e}(C', K_{c_j,k}) \cdot \hat{e}(R_{c_j,k}, C'')^{-1}}{\prod_{i \in I} [\hat{e}(C_i, GPK_{c_j}) \cdot \hat{e}(D_{1,i}, K_{c_j,\rho(i)}) \cdot \hat{e}(D_{2,i}, L_{c_j,k})]^{\delta_i}} \tag{2}$$

$$C' = g^f, C'' = g^{\frac{f}{\beta_k}},$$
$$C_i = g^{a\lambda_i} \cdot ((g^{\vec{v}(i)\rho(i)} \cdot H(\rho(i))^{\gamma_k}))^{-r_i},$$
$$D_{1,i} = g^{\frac{r_i}{\beta_k}}, D_{2,i} = g^{-\frac{\gamma_k}{\beta_k} r_i},$$
$$K_{c_j,k} = g^{\frac{\alpha_k}{z_j}} \cdot g^{a \cdot c_j} \cdot g^{\frac{a}{\beta_k} t_j},$$
$$R_{c_j,k} = g^{a \cdot t_{j,k}}, L_{c_j} = g^{\frac{\beta_k}{z_j} t_{j,k}},$$
$$GPK_{c_j} = g^{c_j}, \text{ and}$$
$$K_{c_j,\rho(i)} = g^{\frac{\beta_k \gamma_k}{z_j} t_{j,k}} \cdot (g^{\rho(i)} \cdot H(\rho(i)))^{\gamma_k \beta_k c_j}.$$

Computing the numerator of Equation ((2)):

$$\begin{aligned} &\hat{e}(C', K_{c_j,k}) \cdot \hat{e}(R_{c_j,k}, C'')^{-1} \\ &= \frac{\hat{e}(g^f, g^{\frac{\alpha_k}{z_j}}) \cdot \hat{e}(g^f, g^{a \cdot c_j}) \cdot \hat{e}(g^f, g^{\frac{a}{\beta_k} t_{j,k}})}{\hat{e}(g^{a \cdot t_{j,k}}, g^{\frac{f}{\beta_k}})} \\ &= \frac{\hat{e}(g,g)^{\frac{f \cdot \alpha_k}{z_j}} \cdot \hat{e}(g,g)^{f \cdot a \cdot c_j} \cdot \hat{e}(g,g)^{\frac{f \cdot a}{\beta_k} t_{j,k}}}{\hat{e}(g,g)^{\frac{f \cdot a}{\beta_k} t_{j,k}}} \\ &= \hat{e}(g,g)^{\frac{f \cdot \alpha_k}{z_j}} \cdot \hat{e}(g,g)^{f \cdot a \cdot c_j}, \end{aligned} \tag{3}$$

Computing the denominator of Equation ((2)):

$$\begin{aligned} &\hat{e}(C_i, GPK_{c_j}) = \hat{e}((g^{a\lambda_i} \cdot ((g^{\vec{v}(i)} \cdot H(\rho(i))^{\gamma_k}))^{-r_i}), g^{c_j}) \\ &= \hat{e}((g^{a\lambda_i}, g^{c_j}) \cdot \hat{e}((((g^{\vec{v}(i)} \cdot H(\rho(i))^{\gamma_k}))^{-r_i}), g^{c_j}) \\ &= \hat{e}(g,g)^{a \cdot \lambda_i \cdot c_j} \cdot \hat{e}(g^{-r_i \cdot \vec{v}(i)\rho(i)}, g^{c_j}) \cdot \hat{e}(H(\rho(i))^{-r_i \cdot \gamma_k}, g^{c_j}) \\ &= \hat{e}(g,g)^{a \cdot \lambda_i \cdot c_j} \cdot \hat{e}(g,g)^{-r_i \cdot c_j \cdot \vec{v}(i)\rho(i)} \cdot \hat{e}(H(\rho(i)), g)^{-r_i \cdot c_j \cdot \gamma_k} \\ &= \frac{\hat{e}(g,g)^{a \cdot \lambda_i \cdot c_j}}{\hat{e}(g,g)^{r_i \cdot c_j \cdot \vec{v}(i)\rho(i)} \cdot \hat{e}(H(\rho(i)), g)^{r_i \cdot c_j \cdot \gamma_k}}, \end{aligned} \tag{4}$$

Let us consider the hash $H(\rho(i))$. Let $g^r = H(\rho(i)) \in \langle g \rangle$ and g be generator of $\mathbb{G}$. Given g and g^r, finding r is very hard because it is DL problem. Let $(x, y) \in \langle g \rangle$ for (x, y) as an elliptic curve point. If we let $x = \rho(i)$, then finding $y^2 = ax^3 + b$ modulo q is easy and can be found; however, in our case, it will be $y^2 = ax^3 + b$ modulo N which is very har because it is square problem.

In addition, obtaining the first part of the denominator for Equation (2): $(\hat{e}(g,g)^{a\cdot\lambda_i\cdot c_j} \cdot \hat{e}(g^{-r_i\cdot\vec{v}(i)\rho(i)}, g^{c_j}) \cdot \hat{e}(H(\rho(i))^{-r_i\cdot\gamma_k}, g^{c_j}))$ is complex. It is based on the extend-gcd, so the attacker cannot obtain it.

$$\begin{aligned}
\hat{e}(D_{1,i}, K_{c_j,\rho(i)}) &= \hat{e}(g^{\frac{r_i}{\beta_k}}, (g^{\frac{\beta_k\gamma_k}{z_j}t_{j,k}} \cdot (g^{\rho(i)} \cdot H(\rho(i)))^{\gamma_k\beta_k c_j})) \\
&= \hat{e}(g^{\frac{r_i}{\beta_k}}, g^{\frac{\beta_k\gamma_k}{z_j}t_{j,k}}) \cdot \hat{e}(g^{\frac{r_i}{\beta_k}}, (g^{\rho(i)} \cdot H(\rho(i)))^{\gamma_k\beta_k c_j}) \\
&= \hat{e}(g,g)^{\frac{r_i\cdot\gamma_k}{z_j}t_{j,k}} \cdot \hat{e}(g^{\frac{r_i}{\beta_k}}, g^{\rho(i)\cdot\gamma_k\beta_k c_j}) \cdot \hat{e}(g^{\frac{r_i}{\beta_k}}, (H(\rho(i)))^{\gamma_k\beta_k c_j}) \\
&= \hat{e}(g,g)^{\frac{r_i\cdot\gamma_k}{z_j}t_{j,k}} \cdot \hat{e}(g,g)^{r_i\cdot\rho(i)\cdot\gamma_k\cdot c_j} \cdot \hat{e}(g, H(\rho(i)))^{r_i\cdot\gamma_k\cdot c_j},
\end{aligned} \tag{5}$$

In addition, obtaining the second part of the denominator for Equation (2) is hard because it belongs to the DL problem and square problem.

$$\begin{aligned}
\hat{e}(D_{2,i}, L_{c_j,k}) &= \hat{e}(g^{-\frac{\gamma_k}{\beta_k}r_i}, g^{\frac{\beta_k}{z_j}t_{j,k}}) \\
&= \hat{e}(g,g)^{\frac{-r_i\cdot\gamma_k}{z_j}t_{j,k}} = \frac{1}{\hat{e}(g,g)^{\frac{r_i\cdot\gamma_k}{z_j}t_{j,k}}}
\end{aligned} \tag{6}$$

After that, the denominator of Equation (2) can be calculated using Equations (4)–(6):

$$\begin{aligned}
&\hat{e}(C_i, GPK_{c_j}) \cdot \hat{e}(D_{1,i}, K_{c_j,\rho(i)}) \cdot \hat{e}(D_{2,i}, L_{c_j,k}) \\
&= \frac{\hat{e}(g,g)^{a\cdot\lambda_i\cdot c_j} \cdot \hat{e}(g,g)^{\frac{r_i\cdot\gamma_k}{z_j}t_{j,k}} \cdot \hat{e}(g,g)^{r_i\cdot\gamma_k\cdot c_j} \cdot \hat{e}(g, H(\rho(i)))^{r_i\cdot\gamma_k\cdot c_j}}{\hat{e}(g,g)^{r_i\cdot c_j\cdot\vec{v}(i)} \cdot \hat{e}(H(\rho(i)), g)^{r_i\cdot c_j\cdot\gamma_k} \cdot \hat{e}(g,g)^{\frac{r_i\cdot\gamma_k}{z_j}t_{j,k}}}
\end{aligned} \tag{7}$$

$$\begin{aligned}
&\hat{e}(C_i, GPK_{c_j}) \cdot \hat{e}(D_{1,i}, K_{c_j,\rho(i)}) \cdot \hat{e}(D_{2,i}, L_{c_j,k}) \\
&= \frac{\hat{e}(g,g)^{a\cdot\lambda_i\cdot c_j} \cdot \hat{e}(g,g)^{r_i\cdot\rho(i)\cdot\gamma_k\cdot c_j}}{\hat{e}(g,g)^{r_i\cdot c_j\cdot\vec{v}(i)\rho(i)}} \\
&= \frac{\hat{e}(g,g)^{a\cdot\lambda_i\cdot c_j\cdot r_i\cdot\rho(i)\cdot\gamma_k\cdot c_j}}{\hat{e}(g,g)^{r_i\cdot c_j\cdot\vec{v}(i)\rho(i)}} \\
&= \frac{\hat{e}(g,g)^{a\cdot\lambda_i\cdot c_j\cdot\gamma_k}}{\hat{e}(g,g)^{\vec{v}(i)}}
\end{aligned} \tag{8}$$

$a, \gamma_k, \vec{v}(i) \in \mathbb{Z}_q^*$ are three randomly chosen constants,

$$\hat{e}(C_i, GPK_{c_j}) \cdot \hat{e}(D_{1,i}, K_{c_j,\rho(i)}) \cdot \hat{e}(D_{2,i}, L_{c_j,k}) = \hat{e}(g,g)^{a\cdot\lambda_i\cdot c_j} \tag{9}$$

By substituting in Equation (2) using Equations (3) and (9):

$$\begin{aligned}
DT &= \frac{\hat{e}(g,g)^{f\cdot a\cdot c_j\cdot N_A} \prod_{k=1}^{N_A} \hat{e}(g,g)^{\frac{f\cdot\alpha_k}{z_j}}}{\hat{e}(g,g)^{a\cdot c_j\cdot N_A\cdot\sum_{i\in I}\delta_i\lambda_i}} \\
&= \frac{\hat{e}(g,g)^{f\cdot a\cdot c_j\cdot N_A} \prod_{k=1}^{N_A} \hat{e}(g,g)^{\frac{f\cdot\alpha_k}{z_j}}}{\hat{e}(g,g)^{a\cdot c_j\cdot N_A\cdot f}} \\
&= \prod_{k=1}^{N_A} \hat{e}(g,g)^{\frac{\alpha_k}{z_j}f}
\end{aligned} \tag{10}$$

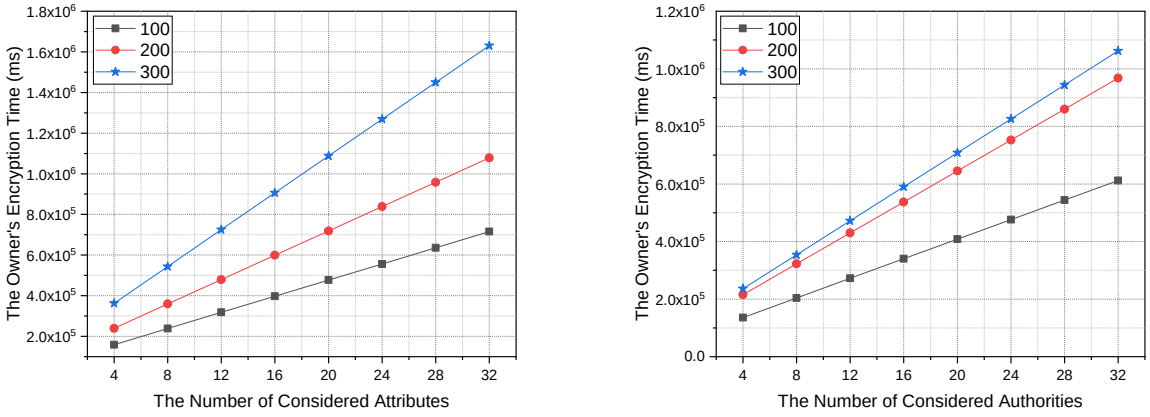
We can find that $DT = \prod_{k=1}^{N_A} \hat{e}(g,g)^{\frac{\alpha_k}{z_j} f}$ is successfully generated. It should be mentioned that if the user does not satisfy the previously mentioned two conditions, once the DT is successfully generated, the decryption algorithm can generate the original data securely. This is because recovering the original data is firstly based on generating the decryption token. Then, DT is used to generate ($ck_k = \frac{CT}{DT^{z_j}}$) for each authority. This is based on two secure factors, CT and DT. Since CT and DT are secure, ck_k is secure and cannot be generated by an attacker who failed to successfully generate the decryption token DT. After that, the user can use $\{ck_k\}_{\forall k \in K}$ to recover $\{\mathcal{D}_k\}_{\forall k \in K}$ from $\{CT_k\}_{\forall k \in K}$, which is also securely recovered based on the security of $\{ck_k\}_{\forall k \in K}$. Finally, the original data $\mathcal{D}$ will be recovered using $\{\mathcal{D}_k\}_{\forall k \in K}$, which is also securely recovered based on the security of recovering $\{\mathcal{D}_k\}_{\forall k \in K}$. Therefore, we guarantee that any attacker under the previously defined security model cannot recover the original data. □

5. Performance Analysis

We implemented TR-AC on top of our private cloud environment. It should be mentioned that all the results are based on an average of 30 trials. The performance of TR-AC was measured as follows.

5.1. Organization's Encryption Time

We considered a different number of attributes and authorities to measure the organization's encryption time, as shown in Figure 3.



(**a**) (**b**)

Figure 3. The encryption time in milli-seconds while considering 100, 200, and 300 access requests. Where (**a**) is the owner's encryption time against the number of considered attributes and (**b**) is the owner's encryption time against the number of considered authorities.

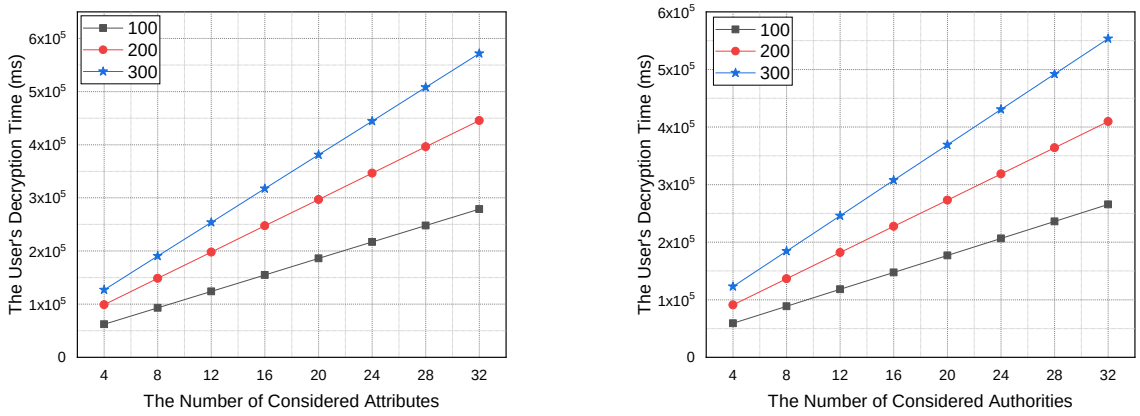
- Figure 3a shows the encryption time according to the number of attributes while considering multiple concurrent access requests to the same authority. The results show that: **i.** The increase in the average encryption time is about 0.0127 seconds when increasing the number of attributes from 4 to 32 while considering 300 concurrent customers; and **ii.** The increase in the average encryption time is about 0.0092 seconds when increasing the number of concurrent customers from 100 to 300 while considering 32 attributes.
- Figure 3b indicates the encryption time according to the number of authorities while considering multiple concurrent access requests to the same authority. The results show that: **i.** The increase in the average encryption time is about 0.0826 seconds when increasing the number of authorities from 4 to 32 while considering 300 concurrent customers; and **ii.** The increase in the average encryption time is about 0.0045

seconds when increasing the number of concurrent customers from 100 to 300 while considering 32 authorities.

Finally, it is clear that the encryption time is reasonable considering the huge stream of concurrent requests on the same authority. Thus, it is accepted by *OiC*. The average encryption time is 0.018 seconds while considering the very tough conditions and huge traffic overhead. This average time is very small and accepted by the data owners.

5.2. Decryption Time

We considered a different number of attributes and authorities to measure the decryption time, as shown in Figure 4.



(**a**) (**b**)

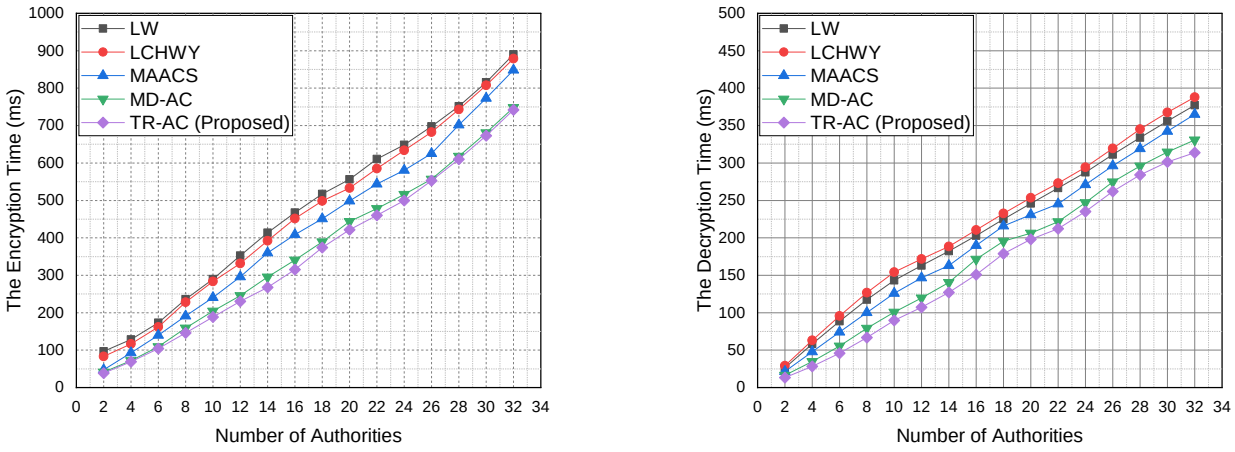
Figure 4. The decryption time in milli-seconds while considering 100, 200, and 300 access requests. Where (**a**) is the user's decryption time against the number of considered attributes and (**b**) is the user's decryption time against the number of considered authorities.

- Figure 4a shows the decryption time according to the number of attributes while considering multiple concurrent access requests to the same authority. The results show that: **i.** The increase in the average decryption time is about 0.00444 seconds when increasing the number of attributes from 4 to 32 while considering 300 concurrent customers; and **ii.** The increase in the average decryption time is about 0.0023 seconds when increasing the number of concurrent customers from 100 to 300 while considering 32 attributes.
- Figure 4b indicates the decryption time according to the number of authorities while considering multiple concurrent access requests to the same authority. The results show that: **i.** The increase in the average decryption time is about 0.0043 seconds when increasing the number of authorities from 4 to 32 while considering 300 concurrent customers; and **ii.** The increase in the average decryption time is about 0.0029 seconds when increasing the number of concurrent customers from 100 to 300 while considering 32 authorities.

Finally, it is clear that the decryption time is reasonable considering the huge stream of concurrent requests to the same authority. The average decryption time is 0.01 seconds when considering the very tough conditions and huge traffic overhead. Thus, it is accepted by the users.

5.3. Comparison with Other Schemes

We compared the time elapsed for encryption and decryption regarding the number of the considered authorities for our TR-AC and four schemes from the related literature (LW [10], LCHWY [11], MAACS [12], and MD-AC [13]), as shown in Figure 5.



(**a**) (**b**)

Figure 5. The time of encryption and decryption regarding different numbers of authorities for our scheme and the related literature (LW [10], LCHWY [11], MAACS [12], MD-AC [13]). Where (**a**) is the encryption time of the related literature and TR-AC (proposed) against the number of considered authorities and (**b**) is the decryption time of the related literature and TR-AC (proposed) against the number of considered authorities.

- Figure 5a shows the encryption time while considering a different number of authorities. The comparisons show that the encryption overhead of those five systems are linear to the scale of access policy in ciphertext; and
- Figure 5b indicates the decryption time while considering a different number of authorities in the system. The comparisons show that the decryption time for all five schemes (LW, LCHWY, MAACS, MD-AC, and TR-AC) is linear to the access policy structure. On the other hand, the time elapsed in TR-AC is shorter than that for the other schemes from the related literature.

6. Conclusions and Future Extensions

TR-AC is designed to facilitate and secure access to sensitive renewable-energy data and energy optimization utilities. We introduced a new token-revocation access control scheme. TR-AC achieved the following goals:

(i) Revoking the malicious users at multiple stages based on the revocation threshold;
(ii) Providing the access to authentic users only;
(iii) The proposed scheme proved to be secure against any attacker that is not authentic, based on the Diffie–Hellman assumption.

By comparing the encryption and decryption times of TR-AC with four well-known schemes (LW, LCHWY, MAACS, and MD-AC), we found that the five schemes' times are linear to the access policy structure. However, the time of TR-AC is shorter than that of LW, LCHWY, MAACS, and MD-AC. Finally, our proposed TR-AC scheme is secure against any attacker who is not authorized. The future extension of TR-AC includes setting a range and time for each permission given to the *UID*.

Funding: This work was funded by the Deputyship for Research & Innovation, Ministry of Education, Saudi Arabia [Project No. INST028].

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: The data used to support the findings of this study are available from the author upon request.

Acknowledgments: The author extends his appreciation to the Deputyship for Research & Innovation, Ministry of Education in Saudi Arabia for funding this research work through the project number INST028.

Conflicts of Interest: The author declares that there is no conflict of interest.

References

1. Basarudin, N.A.; Yeon, A.L.; Yusoff, Z.M. The role of cybersecurity law for sustainability of innovative smart homes (Goal 9). In *Good Governance and the Sustainable Development Goals in Southeast Asia*; Routledge: Abingdon, Oxfordshire, UK, 2022; pp. 110–117.
2. Mojumder, M.; Hasan, R.; Hasanuzzaman, M.; Cuce, E. Prospects and challenges of renewable energy-based microgrid system in Bangladesh: A comprehensive review. *Clean Technol. Environ. Policy* **2022**, *24*, 1987–2009. [CrossRef]
3. Raimundo, R.J.; Rosário, A.T. Cybersecurity in the Internet of Things in Industrial Management. *Appl. Sci.* **2022**, *12*, 1598. [CrossRef]
4. Fagbola, F.I.; Venter, H.S. Smart digital forensic readiness model for shadow IoT devices. *Appl. Sci.* **2022**, *12*, 730. [CrossRef]
5. Vangala, A.; Das, A.K.; Chamola, V.; Korotaev, V.; Rodrigues, J.J. Security in IoT-enabled smart agriculture: Architecture, security solutions and challenges. *Clust. Comput.* **2022**, 1–24. [CrossRef]
6. Lee, C.y. Why do terrorists target the energy industry? A review of kidnapping, violence and attacks against energy infrastructure. *Energy Res. Soc. Sci.* **2022**, *87*, 102459. [CrossRef]
7. Fragkos, G.; Johnson, J.; Tsiropoulou, E.E. Centralized and Decentralized Distributed Energy Resource Access Control Implementation Considerations. *Energies* **2022**, *15*, 6375. [CrossRef]
8. Mohamed, A.K.Y.S.; Auer, D.; Hofer, D.; Küng, J. A systematic literature review for authorization and access control: Definitions, strategies and models. *Int. J. Web Inf. Syst.* **2022**, *18*, 156–180.
9. Rajeh, W. Hadoop Distributed File System Security Challenges and Examination of Unauthorized Access Issue. *J. Inf. Secur.* **2022**, *13*, 23–42. [CrossRef]
10. Lewko, A.; Waters, B. Decentralizing Attribute-Based Encryption. In *Proceedings of the Advances in Cryptology—EUROCRYPT 2011*; Paterson, K.G., Ed.; Springer: Berlin/Heidelberg, Germany, 2011; pp. 568–588.
11. Liu, Z.; Cao, Z.; Huang, Q.; Wong, D.S.; Yuen, T.H. Fully Secure Multi-authority Ciphertext-Policy Attribute-Based Encryption without Random Oracles. In *Proceedings of the Computer Security—ESORICS 2011*; Atluri, V., Diaz, C., Eds.; Springer: Berlin/Heidelberg, Germany, 2011; pp. 278–297.
12. Li, Q.; Ma, J.; Li, R.; Liu, X.; Xiong, J.; Chen, D. Secure, efficient and revocable multi-authority access control system in cloud storage. *Comput. Secur.* **2016**, *59*, 45–59. [CrossRef]
13. Riad, K.; Huang, T.; Ke, L. A dynamic and hierarchical access control for IoT in multi-authority cloud storage. *J. Netw. Comput. Appl.* **2020**, *160*, 102633. [CrossRef]
14. Lampson, B.W.; Alto, P. ACM SIGOPS Operating Systems Review. In *SIGOPS ACM Special Interest Group on Operating Systems*; ACM: New York, NY, USA, 1974; Volume 8, pp. 18–24.
15. Bell, D.; LaPadula, L. *Secure Computer Systems: Mathematical Foundations*; Mitre Corp: Bedford, MA, USA, 1973.
16. Sandhu, R.; Ferraiolo, D.; Kuhn, R. The NIST Model for Role-Based Access Control: Towards a Unified Standard. In Proceedings of the 5th ACM Workshop on Role-Based Access Control, Berlin, Germany, 26–28 July 2000; pp. 47–63.
17. Hu, V.C.; Ferraiolo, D.; Kuhn, R.; Schnitzer, A.; Sandlin, K.; Miller, R.; Scarfone, K. *Guide to Attribute Based Access Control (ABAC) Definition and Considerations*; Special Publication 800-162; National Institute of Standards and Technology (NIST): Gaithersburg, MD, USA, 2014.
18. Harris, S. *Mike Meyers' CISSP(R) Certification Passport*, 1st ed.; McGraw-Hill: New York City, NY, USA, 2002; p. 422.
19. Ferraiolo, D.F.; Sandhu, R.; Gavrila, S.; Kuhn, D.R.; Chandramouli, R. Proposed NIST Standard for Role-Based Access Control. *ACM Trans. Inf. Syst. Secur.* **2001**, *4*, 224–274. [CrossRef]
20. Jin, X.; Krishnan, R.; Sandhu, R. *Data and Applications Security and Privacy XXVI*; Springer: Berlin/Heidelberg, Germany, 2012; Volume 7371, pp. 41–55.
21. Tianyi, Z.; Weidong, L.; Jiaxing, S. An Efficient Role Based Access Control System for Cloud Computing. In Proceedings of the 11th International Conference on: Computer and Information Technology (CIT), Washington, DC, USA, 1 August 2011–2 September 2011; pp. 97–102.
22. Riad, K.; Cheng, J. Adaptive XACML access policies for heterogeneous distributed IoT environments. *Inf. Sci.* **2021**, *548*, 135–152. [CrossRef]
23. Boumezbeur, I.; Zarour, K. Privacy Preservation and Access Control for Sharing Electronic Health Records Using Blockchain Technology. *Acta Inform. Pragensia* **2022**, *11*, 105–122. [CrossRef]

24. Chakraborty, S.; Ray, I. TrustBAC-Integrating Trust Relationships into the RBAC Model for Access Control in Open Systems. In Proceedings of the the ACM Symposium on Access Control Models and Technologies, Lake Tahoe, CA, USA, 7–9 June 2006; pp. 1–6.
25. Boneh, D.; Gentry, C.; Waters, B. Collusion Resistant Broadcast Encryption with Short Ciphertexts and Private Keys. In *Proceedings of the Advances in Cryptology—CRYPTO 2005*; Shoup, V., Ed.; Springer: Berlin/Heidelberg, Germany, 2005; pp. 258–275.