MATRaCAE: Time-Based Revocable Access Control in the IoT

Clémentine Gritti[1], Emanuel Regnath[2] and Sebastian Steinhorst[3]
[1]*INSA Lyon, France*
[2]*Siemens, Munich, Germany*
[3]*Technical University of Munich, Germany*

Keywords: Attribute-Based Encryption, Time-Based Access Control, Direct Revocation, Internet of Things.

Abstract: Internet of Things (IoT) promises a strong connection between digital and physical environments. Nevertheless, this framework comes with security vulnerabilities, due to the heterogeneous nature of devices and the diversity of their provenance. Furthermore, technical constraints (e.g. devices' limited resources) require to lighten the design of the underlying security protocols. Liu et al. presented a system for data access with time-based control and direct user revocation that are beneficial features in IoT. In this paper, we propose an extension of this system, called MATRaCAE, that involves multiple authorities and considers binary time credentials. Doing so, we mitigate the key escrow problem and comes with a better trade-off between key update frequency and number of revoked users, which limited the applicability of Liu et al.'s scheme in IoT. Our solution can be proved secure under the Decisional Bilinear Diffie-Hellman Exponent assumption. Subsequently, we implement and evaluate MATRaCAE to demonstrate its suitability to IoT frameworks.

1 INTRODUCTION

Technologies for the Internet of Things (IoT) have been explored eagerly to offer better efficiency and productivity, but expand vulnerabilities and threats along with technical challenges (Ali et al., 2015). 75 billion devices will be in the IoT world by 2025 and 127 new IoT devices are connected every second to the Internet, yielding the management demanding (Patel et al., 2017). In addition, those devices have various manufacturing origins, not always well defined, and have constrained computing and communication resources (Pham et al., 2016). Those observations make IoT dependability, in particular reliability and availability, challenging (Macedo et al., 2014). Devices continuously collect and exchange a huge amount of data, that is combined and refined through data analytics. IoT has produced more than 500 zettabytes of data per year since 2020, and that number grows exponentially. Developing IoT for capitalizing fresh precious information brings extra security concerns (Hwang, 2015). Consequently, we are interested in developing an efficient access control system for secure data exchanges in IoT networks.

Yet, our solution must be developed carefully. Due to devices' ubiquity and vulnerable high configurations, IoT networks have been involved in many cyber attacks (Pa et al., 2015). Access control in IoT usually implies a centralized architecture, raising single points of failure with unpredictable threats. The large number of devices and dynamicity of IoT networks force to go beyond basic identity assignment techniques as for the Public Key Infrastructure. Trivial key management is restricting since each device should maintain a substantial number of keys to interact with the network. In addition, it is essential to achieve low latency and high reliability. When time-sensitive data is collected, data processing may produce results too late to be useful. For instance, some control decisions in autonomous vehicles require sub-microsecond response times, while industrial control systems need response in tens of microseconds to avoid damage and ensure safety (Mekki et al., 2019). Hence, our access control system should consider time as an essential feature along with effective key management and device revocation to overcome the aforementioned limitations.

In this paper, we introduce *MATRaCAE*[1], a Multi-Authority Time-based Revocable Ciphertext-Policy Attribute-based Encryption scheme for fine-grained access control in IoT, which extends the work proposed in (Liu et al., 2018). An effective balance be-

[1]The full version can be found at https://eprint.iacr.org/2021/140

Gritti, C., Regnath, E. and Steinhorst, S.
MATRaCAE: Time-Based Revocable Access Control in the IoT.
DOI: 10.5220/0012825700003767
Paper published under CC license (CC BY-NC-ND 4.0)
In *Proceedings of the 21st International Conference on Security and Cryptography (SECRYPT 2024)*, pages 274-285
ISBN: 978-989-758-709-2; ISSN: 2184-7711

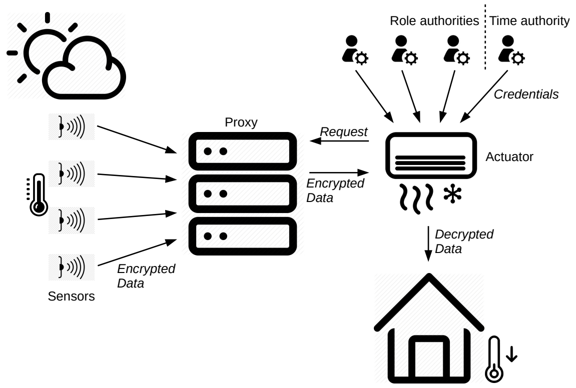

Figure 1: In a smart home, multiple temperature sensors are scattered and an actuator adjusts temperature in response to sensors' collected data.

tween key updates and device revocation permits to offer security guarantees against the above risks while satisfying IoT dependability. Specifically, an access control is built on top of devices' role credentials (e.g. sensing temperature), allowing to share collected data securely within an IoT network. Multiple authorities are in charge of distributing key material to devices, averting the key escrow problem. Direct device revocation keeps sensitive data protected even when a device's secret key is compromised. Moreover, time credentials are used in addition to role credentials, emphasizing the ephemeral value of shared data, while avoiding recurrent communication between devices and authorities. Having short time periods of access allows faster expiration of corrupt device keys, improving device management in IoT networks.

IoT Use Case. Figure 1 illustrates an example of our access control system in a smart home. Several temperature sensors are scattered in a house. They collect temperature values once every few minutes. They encrypt their time-sensitive data according to an access policy, containing roles and time periods. An actuator is connected to these sensors (possibly indirectly, via a gateway). The actuator has received role and time credentials from multiple authorities. Having limited storage capacity, the sensors upload their encrypted data to a proxy (e.g. a cloud server), seen as an intermediary between sensors and actuator. Within the rest of the paper, we assume that the proxy exists and is intimately linked to sensors, hence we omit to mention it explicitly. The actuator sends requests to the proxy for access to sensors' data every short time interval, of the order of minutes. The proxy replies to the actuator's requests by forwarding the encrypted collected data. The actuator is able to recover the data in plain if and only if it has been granted with credentials satisfying sensors' assigned access policies. Having the plain data, the actuator adjusts the temperature accordingly.

1.1 Contributions

We propose an extension, called MATRaCAE, of the access control system presented in (Liu et al., 2018) (referred as LYZL) with the following features to better fit in IoT:

• Device access control is built on top of role and time credentials, as in (Liu et al., 2018).

• The participation of multiple role authorities, rather than a unique one as in (Liu et al., 2018), alleviates the key escrow problem. Nonetheless, authorities must uniquely identify devices. Indeed, an unauthorised device must not access data using a credential from a role authority that incorrectly matches a credential from another role authority.

• A novel time-based access control is designed using binary trees, instead of 31-ary trees as in (Liu et al., 2018), to better apply to IoT frameworks. Moving the time structure from 31-ary trees (based on the Gregorian calendar) to 2-ary trees obliges to carefully specify a process for setting time periods. However, the result is more efficient and adaptable to time-sensitive IoT use cases.

• A direct approach based on a publicly available list for device revocation limits damages from compromised devices' secret keys, as in (Liu et al., 2018).

• Asymmetric pairings are chosen, rather than symmetric pairings as in (Liu et al., 2018), to increase the system security and efficiency, as proved in (Guillevic, 2013). Shifting from symmetric pairings to asymmetric pairings incur less versatility in computing components. Indeed, inputs must be ordered to ensure that pairing calculations are still possible (e.g. successful decryption).

We obtain a better balance between key update and device revocation compared to (Liu et al., 2018), making MATRaCAE suitable for IoT networks. We adapt the Ciphertext-Policy Attribute-Based Encryption (CP-ABE) scheme and security model from (Liu et al., 2018) to follow the multi-authority setting, and prove our scheme secure under the Decisional Bilinear Diffie-Hellman Exponent (BDHE) assumption. We also implement and evaluate MATRaCAE to confirm its dependability in IoT.

1.2 Related Work

Attribute-Based Encryption. Identity-Based Encryption (IBE) (Shamir, 1985) is a public-key cryptographic primitive that uses some unique information about the user identity as her public key. The corresponding secret key is generated by a trusted authority, based on the public key. Attribute-Based Encryption (ABE) (Sahai and Waters, 2005; Bethencourt et al., 2007) is a variant of IBE. The user secret key and ciphertext are dependent upon attributes. The decryption of a ciphertext is possible if and only if the attributes of the key match the attributes of the ciphertext. There are two types of ABE schemes, that are closely related. Their difference comes from the access policy being linked either to the key (KP-ABE) or to the ciphertext (CP-ABE).

Multi-authority ABE schemes have been proposed over the last decade (Rouselakis and Waters, 2015; Datta et al., 2021; Ambrona and Gay, 2023), but without incorporating revocation and time-based mechanisms. On the other hand, several revocable ABE schemes have been presented (Sahai et al., 2012; Yang and Jia, 2014; Liu et al., 2018; Liu et al., 2020; Zhang et al., 2019; Zhang et al., 2022b), but are prone to the key escrow problem. In (Sahai et al., 2012), revocation is made possible through time-related binary trees, but in the case of KP-ABE only. In (Yang and Jia, 2014), revocation, conducted by the attribute authorities, implies to update the secret keys of non-revoked users. In (Liu et al., 2018), a time-based CP-ABE is combined with direct revocation tools. However, time design choices induce ineffective trees. In (Liu et al., 2020; Zhang et al., 2019), direct revocation is provided but the solutions suffer from ineffecient time-based control. In (Zhang et al., 2022b), revocation is possible but key updates must be performed by specific authorities.

Attribute-Based Encryption in IoT. In (Yao et al., 2015), an ABE scheme without any pairing operation is presented to save costs and possibly be used for IoT. (Oualha and Nguyen, 2016) extends (Bethencourt et al., 2007) with a focus on IoT, but precomputed values, generated by a trusted authority, incur extra storage. More recently, IoT access control solutions (Lu et al., 2021; Zhang et al., 2021; Zhang et al., 2023) have been proposed combining ABE with new technologies such as blockchain and lightweight cryptography. However, all the aforementioned works lack of essential features, such as device revocation and key escrow mitigation. Other ABE-based IoT systems (AboDoma et al., 2021; Zhang et al., 2022a; Yan et al., 2023) with revocation and computation outsourcing have been presented. Nevertheless, computation outsourcing requires trust assumptions and key escrow issues are not considered.

2 BUILDING BLOCKS

Multiple Authorities. We enhance LYZL by involving multiple authorities. In (Liu et al., 2018), one fully trusted authority is responsible of generating the key material of users. Such a configuration may be subject to key escrow and single point of failure. By sharing the generation of devices' keys among several authorities, we reduce trust assumptions made on these authorities while enforcing the security of MATRaCAE. We assume that one authority remains honest at all time to keep MATRaCAE secure.

Revocation. Reasons for revocation can be diverse: (i) The device left the IoT network, thus the key should no longer be usable; (ii) The device lost its key and was attributed a new one, hence the old key should no longer be usable; (iii) The device has one of its credentials changed and thus has received a new key with this new credential, and the old key should no longer be usable.

Existing revocation approaches rely on key updates of non-revoked devices or cloud assistance. However, the former does not allow direct revocation while the latter encounters management issues when the number of devices becomes huge. Another approach allows to revoke devices by appending their identities in a public list. Such a list is included in each ciphertext in its latest version. Hence, key update is not needed, avoiding communication burdens. However, the list may grow significantly over time, in particular in large IoT networks. A solution, suggested in (Liu et al., 2018), is to find a balance between the frequency of key updates and the length of the revocation list. This list accepts a maximum number of revoked devices such that, once reached, new keys are distributed and the list is emptied. Key updates are defined such that they happen just before the list is full. A time period is specified with an expiry date such that, once passed, the device is no longer able to access data. Devices' keys are updated with a new time period. If a device is revoked before its key expires, then its identity is added to and kept in the revocation list until the next key update. Note that a device may be revoked definitely from the network, and its key is no longer updated. This direct revocation mechanism is used in MATRaCAE.

Access Tree. In (Liu et al., 2018), a continuous time period is defined during encryption such that only users with time credentials completely covering that time period can decrypt. If a user has time credential "January 2024" while the encryptor has set "from 01 to 15 January 2024", then the former successfully

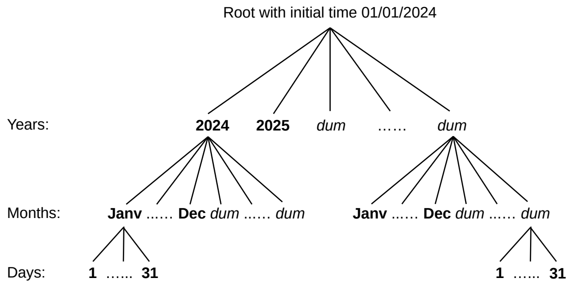

Figure 2: Tree set for 2 years, from “01 January 2024” until “31 December 2025” (included) (Liu et al., 2018). Let ”dum” denote dummy nodes.

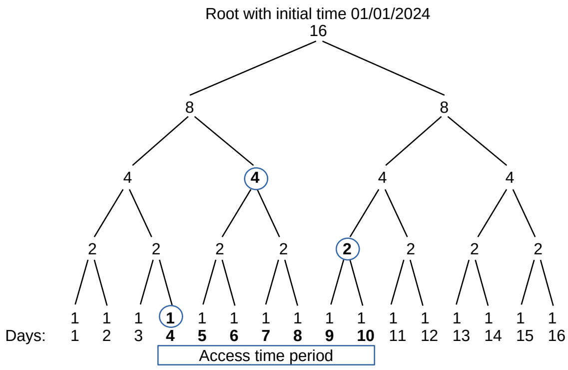

Figure 3: Tree set from “01 January 2024” until “16 January 2024” (included). A time period is defined for 7 days. Keys correspond to nodes with blue-line circles.

decrypts. Such properties are kept when designing MATRaCAE. A *set cover* approach is used to select the minimum number of tree nodes that represent the valid time periods. The root node is implicitly set as a starting time. Each non-root node accounts for a time period such that leaves are days, leaves' parents are months and leaves' grand-parents are years. Let T be the depth of the tree and each node have z children. The time is represented as a z-ary string $\{1,2,\cdots,z\}^{T-1}$ and a time period is denoted with a z-ary element $(\tau_1,\tau_2,\cdots,\tau_\eta)$ for some $\eta < T$. No numerical value is given in (Liu et al., 2018), hence we propose to make some assumptions from the reading. A 2-year interval between two key updates is claimed to be reasonable and time periods are based on year, month and day. We infer that $T = 4$ and $z = 31$ (there are at most 31 days in a month). Thus, the root and nodes representing years have $z = 31$ children, even if intervals are 2 years long and years comprise 12 months. This approach is cumbersome because there are $31-2 = 29$ dummy nodes at the year level, $29*(31-12) = 551$ dummy nodes at the month level and $29*19*31 = 17081$ dummy nodes at the day level. Figure 2 illustrates the above 2-year period example from LYZL.

We now describe the LYZL access tree depicting time intervals. The set cover mechanism allows to find the minimum number of nodes representing the time validity range. Let a validity time range be from “30 November 2024” until “31 December 2025”. The selected nodes in Figure 2 would be the node “30” with parent “November” and grand-parent “2024”, the node “December” with parent “2024”, and the node “2025”. Let η_τ be the number of selected nodes and T be the depth of the tree such that $\eta_\tau < T$. Let $\mathbb{T} = (\tau_1,\tau_2,\cdots,\tau_{\eta_\tau})$ be the set cover representing the time validity range. Following the above example, $\eta_\tau = 3$ such that $\tau_1 =$ “30 November 2024“, $\tau_2 =$ ”December 2024“, and $\tau_3 =$ ”2025“. We have not mentioned the presence of the dummy nodes to not overload the understanding. However, they must be included in the set cover, thus $\eta_\tau >> 3$.

In MATRaCAE, a tree is also used for time-based access control. A time authority manages trees and assigns time intervals as devices' credentials. We choose a binary structure rather than a 31-ary structure, avoiding the presence of dummy nodes. In addition, we focus on short time periods (of the order of days) according to our IoT time-sensitive data scenario. The root defines a starting time and the number of leaves determines the amount of days between two key updates. To keep the tree with a reasonable depth T, the number of leaves must be relatively small. A path from the root to a node is denoted as a string in $\{0,1\}^{T-1}$ where 0 denotes the left child and 1 denotes the right child of a given node. Following the above 2-year period example, our binary tree would have 730 leaves, so a depth $T = 10$. A complete binary tree of depth $T = 10$ has 1023 nodes (including dummies). However, following (Liu et al., 2018), a 31-ary tree with 29791 nodes (including dummies) would be built, making computation and storage costs noticeably worse than ours.

In Figure 3, the tree has depth $T = 5$, hence 16 leaves (thus 16 days). The time interval starts on “01 January 2024” and ends on “16 January 2024” (included). Here, a device receives some time key material for a time validity range of 7 days, from “04 January 2024” until “10 January 2024” (included). The device is given 3 key components as illustrated by blue circles: one for the leaf node representing day 4, one for the grand-parent from day 5 until day 8, and one for the parent for days 9 and 10. Following (Liu et al., 2018), a 31-ary tree of depth $T = 4$ would incur 7 keys, one for each day, for the same time interval.

Similarly to (Liu et al., 2018), an access tree represents time periods in MATRaCAE. Rather than using 31-ary strings $\{1,2,\cdots,31\}^{T-1}$, we consider binary strings $\{0,1\}^{T-1}$ such that a node represented as 0 means going to the left child and as 1 means going to the right child. Let the validity time range be from "04 January 2024" until "10 January 2024". The selected nodes are the ones circled in blue in Figure 3. Here, $\mathbb{T} = (\tau_1, \tau_2, \tau_3)$ where $\tau_1 = (0,0,1,1)$, $\tau_2 = (0,1)$ and $\tau_3 = (1,0,0)$.

Bilinear Pairing and Group. Let $\mathbb{G}_1$, $\mathbb{G}_2$ and $\mathbb{G}_T$ be multiplicative cyclic groups of prime order p. A pairing e is a map $e : \mathbb{G}_1 \times \mathbb{G}_2 \rightarrow \mathbb{G}_T$ such that: (i) Given $g_1 \in \mathbb{G}_1$, $g_2 \in \mathbb{G}_2$ and $a,b \in \mathbb{Z}_p$, $e(g_1^a, g_2^b) = e(g_1,g_2)^{ab}$ (bilinearity); (ii) There exist $g_1 \in \mathbb{G}_1$ and $g_2 \in \mathbb{G}_2$ such that $e(g_1,g_2) \neq 1_{\mathbb{G}_T}$ (non-degeneracy); (iii) There exists an efficient algorithm to compute $e(g_1,g_2)$ for all $g_1 \in \mathbb{G}_1$ and $g_2 \in \mathbb{G}_2$ (computability).

Given as input a security parameter 1^λ, the algorithm Gen outputs the tuple $(p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e)$ where $\mathbb{G}_1, \mathbb{G}_2$, $\mathbb{G}_T$ are multiplicative cyclic groups of prime order p and $e : \mathbb{G}_1 \times \mathbb{G}_2 \rightarrow \mathbb{G}_T$ is a pairing.

When $\mathbb{G}_1 = \mathbb{G}_2$, the pairing is *symmetric*, as in (Liu et al., 2018). When $\mathbb{G}_1 \neq \mathbb{G}_2$, the pairing is *asymmetric*. Designing a scheme with asymmetric pairings (rather than symmetric pairings) offers better performances and improves the security (Guillevic, 2013). MATRaCAE extends LYZL with asymmetric pairings to enhance its security in IoT.

Decisional q-BDHE Assumption. Given $\vec{P} = (g_1, g_1^s, g_1^a, \cdots, g_1^{a^q}, g_1^{a^{q+2}}, \cdots, g_1^{a^{2q}}, g_2, g_2^s, g_2^a, \cdots, g_2^{a^q}, g_2^{a^{q+2}}, \cdots, g_2^{a^{2q}}) \in \mathbb{G}_1^{2q+1} \times \mathbb{G}_2^{2q+1}$ and $Q \in \mathbb{G}_T$, where $s,a \in \mathbb{Z}_p$, $g_1 \in \mathbb{G}_1$ and $g_2 \in \mathbb{G}_2$, the Decisional q-Bilinear Diffie-Hellman Exponent (BDHE) problem is defined as to decide whether $Q = e(g_1,g_2)^{sa^{q+1}}$ or a random element in $\mathbb{G}_T$. The security of MATRaCAE relies on the Decisional q-BDHE assumption.

Access Structure and Linear Secret Sharing. Let $\mathcal{P} = \{P_1, P_2, \cdots, P_n\}$ be a set of parties. A collection $\mathbb{C} \subseteq 2^{\mathcal{P}}$ is monotone if for all A, B, such that $A \in \mathbb{C}$ and $A \subseteq B$ then $B \in \mathbb{C}$. An access structure is a collection $\mathbb{C} \subseteq 2^{\mathcal{P}} \setminus \{\emptyset\}$. The sets in $\mathbb{C}$ are said to be *authorized*. In MATRaCAE, this structure represents access policies and is used for role-based access control.

A Secret Sharing Scheme (SSS) Π over a set of parties $\mathcal{P}$ is a Linear SSS (LSSS) if the following conditions hold (Beimel, 1996): (i) The shares of the parties form a vector over $\mathbb{Z}_p$; (ii) There are a $l \times \nu$ matrix M and a function ρ that maps the i-th row, for $i \in [1,l]$, to an associated party $\rho(i)$. Let $s \in \mathbb{Z}_p$ be a secret to be shared, and $\gamma_2, \cdots, \gamma_\nu$ be random exponents from $\mathbb{Z}_p$. Let $\vec{v} = (s, \gamma_2, \cdots, \gamma_\nu)$ be a column vector and $M\vec{v}$ be the vector of l shares of the secret s according to Π such that the share $(M\vec{v})_i$ belongs to party $\rho(i)$.

We define the linear reconstruction property as follows. Let Π be an LSSS for an access structure $\mathbb{C}$, $S \in \mathbb{C}$ be an authorized set and $I = \{i; \rho(i) \in S\} \subset [1,l]$. There exist constants $\{\omega_i \in \mathbb{Z}_p\}_{i \in I}$ such that, if $\{\lambda_i\}$ are valid shares of any secret s according to Π, then $\sum_{i \in I} \omega_i \lambda_i = s$. The constants ω_i can be found in time polynomial in the size of M. Moreover, for any unauthorized set $S \notin \mathbb{C}$, the secret s should be information theoretically hidden from the parties in S. Let the vector $(1,0,0,\cdots,0)$ be the *target* vector for LSSS. Given an authorized set of rows I in the matrix M, the target vector is in the span of I. On the other side, given an unauthorized set of rows I, the target vector is not in the span of the rows of I. Also, there is a vector $\vec{w}$ such that $\vec{w}(1,0,0,\cdots,0) = -1$ and $\vec{w}M_i = 0$ for all $i \in I$. In MATRaCAE, the LSSS matrix has rows labeled by role attributes.

Role Attributes and Indexation. In (Liu et al., 2018), an attribute universe is associated with the unique authority such that attributes are all different. In MATRaCAE, each role authority A_k has its own attribute universe $\mathcal{U}_k$ such that the union of all the attribute universes forms the network universe $\mathcal{U} = \cup_{A_k} \mathcal{U}_k$. Attribute universes are all disjoint by defining attributes uniquely as follows. Let a role be "temperature" and two authorities refer to "Room A" and "Room B" respectively. Hence, the two attributes are determined uniquely as "RoomA||temperature" and "RoomB||temperature" respectively. The authorities define their own universes and assign role credentials based on those universes in devices' keys. Key updates w.r.t. roles are not frequent (e.g. "temperature" remains for a dedicated sensor but its location may change). Devices can decrypt data if the access policy contains their role attributes.

Indexation works as follows. Let N be the number of role authorities. Let A_k be the role authority with universe $\mathcal{U}_k$ containing U_k attributes, for $k \in [1,N]$. Attribute indices in $\mathcal{U}_k$ associated with A_k are $(\sum_{j=1}^{k-1} U_j + 1), \cdots, (\sum_{j=1}^{k-1} U_j + U_k)$. To simplify the reading, let $k||i = (\sum_{j=1}^{k-1} U_j + i)$ for $i \in [1, U_k]$. Let $I = \{i; \rho(i) \in S\} \subseteq [1,l]$ and $\{\omega_i \in \mathbb{Z}_p\}_{i \in I}$ be the set of constants such that if the set $\{\lambda_i\}$ contains valid shares of a value s according to the matrix M, then $\sum_{i \in I} \omega_i \lambda_i = s$, as defined previously. Let $\mathcal{A}$ be the set of role authorities whose attributes are in the access structure. Let $\pi : k \rightarrow \pi(i)$ be defined as $\exists!(A_k \in \mathcal{A}, j \in [1, U_k])$ such that $\rho(i) = k||j$. This surjective function exists since each attribute is defined uniquely in the network universe $\mathcal{U} = \cup_{A_k} \mathcal{U}_k$. An attribute in $\mathcal{U}$ is uniquely controlled by one authority A_k. To explain the functionality of the function π, there exists a publicly computable function

$F_\pi : \mathcal{U} \rightarrow \mathcal{A}_k$ that maps one attribute to a specific role authority (Rouselakis and Waters, 2015). From this mapping, let a second labeling of rows be defined in the access structure $((M,\rho),\rho')$ such that it maps rows to attributes through $\rho(\cdot) = F_\pi(\rho'(\cdot))$.

3 MATRaCAE

An overview of MATRaCAE is given in Figure 4. MATRaCAE is composed of seven algorithms run over three phases:

Initialization. This phase, run only once, sets the system parameters.

• $\mathsf{Setup}(1^\zeta, R) \rightarrow PP$. The algorithm Setup generates the public parameters made available to all authorities and devices. Let $R-1$ be the maximum number of revoked devices. On inputs the security parameter 1^ζ and R, the algorithm Setup outputs the public parameters PP. First, run the algorithm Gen and obtain two bilinear groups $\mathbb{G}_1, \mathbb{G}_2$ of prime order p with generators g_1 and g_2 respectively, along with a third group $\mathbb{G}_T$ of prime order p and a pairing $e : \mathbb{G}_1 \times \mathbb{G}_2 \rightarrow \mathbb{G}_T$. Pick at random $\delta, \alpha_1, \cdots, \alpha_R \in_R \mathbb{Z}_p$. Set $\vec{\alpha} = (\alpha_1, \cdots, \alpha_R)^\top$ and $\vec{F} = g_1^{\vec{\alpha}} = (g_1^{\alpha_1}, \cdots, g_1^{\alpha_R})^\top = (f_1, \cdots, f_R)^\top$. The public parameters are $PP = (p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e, g_1, g_1^\delta, g_2, \vec{F})$.

• $\mathsf{RAKeyGen}(PP, U_k) \rightarrow (PK_k, SK_k)$. The algorithm RAKeyGen generates the public and secret keys of each role authority. Let U_k be the number of role attributes in the universe $\mathcal{U}_k$ associated with the role authority A_k. On inputs the public parameters PP and U_k, the algorithm RAKeyGen outputs the public key PK_k and the secret key SK_k of A_k. Pick at random $\kappa_k \in_R \mathbb{Z}_p$ and $h_{k||1}, \cdots, h_{k||U_k} \in_R \mathbb{G}_1$ (these elements $h_{k||i}$ will be used for role-based access control w.r.t. A_k). The public key is $PK_k = (e(g_1,g_2)^{\kappa_k}, h_{k||1}, \cdots, h_{k||U_k})$ and the secret key is $SK_k = \kappa_k$.

• $\mathsf{TAKeyGen}(PP, T) \rightarrow (PK, SK)$. The algorithm TAKeyGen generates the public and secret keys of the time authority. Let T be the depth of the binary tree associated with the time authority B. The time is represented as a binary string $\{0,1\}^{T-1}$. On inputs the public parameters PP and T, the algorithm TAKeyGen outputs the public key PK and the secret key SK of B. Pick at random $\sigma \in_R \mathbb{Z}_p$ and $V_0, V_1, \cdots, V_T \in_R \mathbb{G}_1$ (the elements V_j will be used for time-based access control w.r.t. B). The public key is $PK = (e(g_1,g_2)^\sigma, V_0, V_1, \cdots, V_T)$ and the secret key is $SK = \sigma$.

Key Generation. The authorities create devices' key material. For $1 \leq k \leq N$, the role authority A_k generates keys based on roles, defined in the universe $\mathcal{U}_k$. Role-based key updates for non-revoked devices are run occasionally (e.g. every few months). The time authority B generates keys based on time intervals, denoted as T_{ID} where ID is the device identity. Time-based key updates are run more frequently (e.g. every few days/weeks).

• $\mathsf{RUKeyGen}(PP, (PK_k, SK_k), ID, S_{ID,k}) \rightarrow RSK_{ID,k}$. The algorithm RUKeyGen generates the secret key of the device w.r.t. its roles. Let $S_{ID,k}$ be the role attribute set of a device with identity ID and associated with the role authority A_k. Let $k||x \in S_{ID,k}$ denote the attribute uniquely defined in the network universe $\mathcal{U} = \cup_{A_k} \mathcal{U}_k$ by determining the associated authority A_k and the role x within $\mathcal{U}_k$. On inputs the public parameters PP, the public and secret keys PK_k and SK_k of A_k, ID and $S_{ID,k}$, the algorithm RUKeyGen outputs the secret key $RSK_{ID,k}$ of the device with identity ID, role attribute set $S_{ID,k}$ and associated with A_k. First, pick at random $u_k, t_k \in_R \mathbb{Z}_p$. Then, compute the following:

$$\begin{aligned} D_{k,0} &= g_2^{t_k} \\ D'_{k,0} &= g_2^{u_k} \\ D_{k,1} &= g_1^{\kappa_k} g_1^{\delta t_k} f_1^{u_k} = g_1^{\kappa_k} g_1^{\delta t_k} g_1^{\alpha_1 u_k} \\ K_{k,x} &= h_{k||x}^{t_k} \text{ for } k||x \in S_{ID,k} \\ F_{k,i} &= (f_1^{-ID^{i-1}} f_i)^{u_k} \text{ for } i \in [2,R] \end{aligned}$$

The secret key is $RSK_{ID,k} = (D_{k,0}, D'_{k,0}, D_{k,1}, \{K_{k,x}\}_{k||x \in S_{ID,k}}, \{F_{k,i}\}_{i \in [2,R]})$ and includes a description of $S_{ID,k}$.

• $\mathsf{TUKeyGen}(PP, (PK, SK), ID, T_{ID}) \rightarrow TSK_{ID}$. The algorithm TUKeyGen generates the secret key of the device w.r.t. its access time period. Let T_{ID} be the time validity range of the device with identity ID and associated with the time authority B. On inputs the public parameters PP, the public and secret keys PK and SK of B, ID and T_{ID}, the algorithm TUKeyGen outputs the secret key TSK_{ID} of the device with identity ID, time validity range T_{ID} and associated with B. Let $\mathbb{T}$ be the set cover representing T_{ID} which consists of time elements $\tau = (\tau_1, \cdots, \tau_{\eta_\tau}) \in \{0,1\}^{\eta_\tau}$ where $\eta_\tau < T$ for any $\tau \in \mathbb{T}$. First, pick at random $\beta, v_\tau \in_R \mathbb{Z}_p$ for $\tau \in \mathbb{T}$. Then, compute the following:

$$\begin{aligned} D_{0,\tau} &= g_2^{v_\tau} \text{ for } \tau \in \mathbb{T} \\ D_{1,\tau} &= g_1^\sigma f_1^\beta (V_0 \prod_{j=1}^{\eta_\tau} V_j^{\tau_j})^{v_\tau} \text{ for } \tau \in \mathbb{T} \\ D_2 &= g_2^\beta \\ L_{j,\tau} &= V_j^{v_\tau} \text{ for } j \in [\eta_\tau + 1, T] \text{ and } \tau \in \mathbb{T} \end{aligned}$$

Phase	Algorithm	Role Authority	Time Authority	Sensor	Actuator
Initialization	Setup	PP	PP	PP	PP
	RAKeyGen	PK_k, SK_k			
	TAKeyGen		PK, SK		
Key Generation	RUKeyGen	$\xrightarrow{RSK_{ID,k}}$			
	TUKeyGen		$\xrightarrow{TSK_{ID}}$		
Encryption-Decryption	Encrypt			$\xrightarrow{CT}$	
	Decrypt				m

Figure 4: Overview of MATRaCAE.

$$E_i = (f_1^{-ID^{i-1}} f_i)^{\beta} \text{ for } i \in [2,R]$$

The secret key is $TSK_{ID} = (\{D_{0,\tau}, D_{1,\tau}\}_{\tau\in\mathbb{T}}, D_2, \{L_{j,\tau}\}_{j\in[\eta_\tau+1,T],\tau\in\mathbb{T}}, \ \{E_i\}_{i\in[2,R]})$ and includes a description of T_{ID}.

Encryption-Decryption. Let an access policy be (M,ρ) where M is a $l \times \nu$ matrix and the function ρ associates rows of the matrix M to role attributes. Let a decryption time period be T_{dec}. A device encrypts collected data m based on (M,ρ) and T_{dec}, along with the up-to-date revocation list $\mathcal{R}$ (with up to $R-1$ revoked devices), resulting in a ciphertext CT. Another device, granted with role and time credentials satisfying (M,ρ) and T_{dec} respectively, successfully decrypts CT and recovers m.

• Encrypt$(PP, \{PK_k\}_{A_k\in\mathcal{A}}, PK, m, \mathcal{R}, (M,\rho), T_{dec}) \rightarrow CT$. The algorithm Encrypt generates a ciphertext of the message m. Let $\mathcal{A}$ be the set of role authorities whose role attributes are in the access policy. Let m be the message to be encrypted. Let $\mathcal{R} = (ID_1, \cdots, ID_r)$ be the revocation list containing $r < R$ revoked devices. Let (M,ρ) be a LSSS access structure, defining the role access policy. Let T_{dec} be the decryption time period of the ciphertext. On inputs the public parameters PP, the public keys PK_k of the role authorities $A_k \in \mathcal{A}$, the public key PK of the time authority B, m, $\mathcal{R}$, (M,ρ) and T_{dec}, the algorithm Encrypt outputs a ciphertext CT. Let $\tau_{dec} = (\tau_1, \cdots, \tau_{\eta_{dec}}) \in \{0,1\}^{\eta_{dec}}$ be the binary representation of T_{dec}, where $\eta_{dec} < T$. First, choose a secret s from $\mathbb{Z}_p$ and pick at random $\gamma_2, \cdots, \gamma_\nu \in_R \mathbb{Z}_p$. Set the vector $\vec{v} = (s, \gamma_2, \cdots, \gamma_\nu)$. Then, for $i \in [1,l]$, compute $\lambda_i = \langle \vec{v}, M_i \rangle$, where M_i is the i-th row of M. Let $\mathcal{F}_{\mathcal{R}}(Z) = (Z - ID_1)\cdot(Z - ID_2)\cdots(Z - ID_r) = y_1 + y_2 Z + \cdots + y_r Z^{r-1} + y_{r+1} Z^r$ be a polynomial defining the revocation list. If $r+1 < R$, then set the coefficients $y_{r+2}, \cdots, y_R$ equal to 0. Then, compute the following:

$$C_0 = m \cdot e(g_1,g_2)^{\sigma s} \cdot \prod_{A_k\in\mathcal{A}} e(g_1,g_2)^{\kappa_k s}$$

$$C_0' = g_2^s$$

$$C_0'' = (f_1^{y_1} \cdots f_R^{y_R})^s$$

$$C_0''' = (V_0 \prod_{j=1}^{\eta_{dec}} V_j^{\tau_j})^s$$

$$C_i = g_1^{\delta\lambda_i} h_{\rho(i)}^{-s} \text{ for } i \in [1,l]$$

The ciphertext is $CT = (C_0, C_0', C_0'', C_0''', \{C_i\}_{i\in[1,l]}, (M,\rho))$ and includes descriptions of T_{dec} and $\mathcal{A}$.

• Decrypt$(PP, CT, \mathcal{R}, \{RSK_{ID,k}\}_{A_k\in\mathcal{A}}, TSK_{ID}) \rightarrow m/\perp$. The algorithm Decrypt attempts to recover the message m from the ciphertext using appropriate secret parameters. On inputs the public parameters PP, the ciphertext CT, the revocation list $\mathcal{R}$, the role secret keys $RSK_{ID,k}$ of the device with identity ID and associated with $A_k \in \mathcal{A}$ and the time secret key TSK_{ID} of the device with identity ID and associated with B, the algorithm Decrypt outputs m or $\perp$.

Let $\vec{X} = (1, ID, \cdots, ID^{R-1})$ for the identity ID and $\vec{Y} = (y_1, \cdots, y_R)$ where the exponents y_i have been defined during the encryption phase. Hence, $\langle \vec{X}, \vec{Y} \rangle = y_1 + y_2 ID + \cdots + y_r ID^{r-1} + y_{r+1} ID^r = \mathcal{F}_{\mathcal{R}}(ID)$. If $r+1 < R$, then the coefficients $y_{r+2}, \cdots, y_R$ are equal to 0. Let $S_{ID} = \cup_{A_k\in\mathcal{A}} S_{ID,k}$ be the disjoint union of all the role attribute sets $S_{ID,k}$ of the device with identity ID and associated with $A_k \in \mathcal{A}$. Let τ_{dec} be the binary representation for the decryption time period T_{dec} and $\mathbb{T}$ be the set cover representing the time validity range T_{ID}. Let us define the following conditions:

• *Insufficient roles attributes:* S_{ID} does not satisfy (M,ρ);

• *Revoked device:* $ID \in \mathcal{R}$, that is $\langle \vec{X}, \vec{Y} \rangle = \mathcal{F}_{\mathcal{R}}(ID) = 0$;

• *Invalid access time period:* T_{dec} is not completely covered in T_{ID}, that is τ_{dec} and all its prefixes are not in $\mathbb{T}$.

If any of the above conditions occurs, then output $\perp$ and abort. Otherwise, since $\langle \vec{X}, \vec{Y} \rangle \neq 0$, compute the following:

$$F_k = \prod_{i=2}^{R} F_{k,i}^{y_i} = (f_1^{-\langle \vec{X},\vec{Y} \rangle} \prod_{i=1}^{R} f_i^{y_i})^{u_k}$$

$$\xi_{k,1} = \left(\frac{e(F_k, C_0')}{e(C_0'', D_{k,0}')}\right)^{\frac{-1}{\langle \vec{X}, \vec{Y}\rangle}} = e(g_1, g_2)^{\alpha_1 s u_k}$$

$$E = \prod_{i=2}^{R} E_i^{y_i} = (f_1^{-\langle \vec{X}, \vec{Y}\rangle} \prod_{i=1}^{R} f_i^{y_i})^{\beta}$$

$$\xi_1' = \left(\frac{e(E, C_0')}{e(C_0'', D_2)}\right)^{\frac{-1}{\langle \vec{X}, \vec{Y}\rangle}} = e(g_1, g_2)^{\alpha_1 s \beta}$$

Let $I \subseteq [1, l]$ be defined as $\{i; \rho(i) \in S_{ID}\}$ and $\{\omega_i \in \mathbb{Z}_p\}_{i \in I}$ be the set of constants such that if the set $\{\lambda_i\}$ contains valid shares of a value s according to the matrix M, then $\sum_{i \in I} \omega_i \lambda_i = s$. In addition, there is a surjective function from I to $\mathcal{A}$ determined as follows. Let $\pi : k \rightarrow \pi(i)$ be defined as $\exists!(A_k \in \mathcal{A}, j \in [1, U_k])$ such that $\rho(i) = k||j$. Such function exists since each attribute is defined uniquely in the network universe $\mathcal{U} = \cup_{A_k} \mathcal{U}_k$. Then, compute:

$$\xi_2 = \prod_{i \in I} \left(e(C_i, D_{\pi(i),0}) \cdot e(K_{\rho(i)}, C_0')\right)^{\omega_i} = \prod_{A_k \in \mathcal{A}} e(g_1, g_2)^{\delta s t_k}$$

If $\tau_{dec} = (\tau_1, \cdots, \tau_{\eta_{dec}}) \in \mathbb{T}$, then $D_{1,\tau_{dec}}$ should be one component of the secret key TSK_{ID}. Otherwise, let $\tau_{dec}' = (\tau_1, \cdots, \tau_{\eta_{dec}'})$ denote the prefix such that $\eta_{dec}' < \eta_{dec}$ and $\tau_{dec}' \in \mathbb{T}$. Then, derive a key component $D_{1,\tau_{dec}}$ from TSK_{ID} with respect to τ_{dec}' by calculating $D_{1,\tau_{dec}} = D_{1,\tau_{dec}'} \prod_{j=\eta_{dec}'+1}^{\eta_{dec}} L_{j,\tau_{dec}'}^{\tau_j}$ and set $\tau_{dec} = \tau_{dec}'$. Finally, recover the message m as follows:

$$m = C_0 \cdot \xi_2 \cdot \frac{e(D_{0,\tau_{dec}}, C_0''') \cdot \xi_1'}{e(D_{1,\tau_{dec}}, C_0')} \cdot \prod_{A_k \in \mathcal{A}} \frac{\xi_{k,1}}{e(D_{k,1}, C_0')}$$

Correctness. We first calculate F_k. Implicitly, the device must not have been revoked to get a correct result $F_k = \prod_{i=2}^{R} F_{k,i}^{y_i} = (f_1^{-\langle \vec{X}, \vec{Y}\rangle} \prod_{i=1}^{R} f_i^{y_i})^{u_k}$. Using the above result, we calculate $\xi_{k,1} = \left(\frac{e(F_k, C_0')}{e(C_0'', D_{k,0}')}\right)^{\frac{-1}{\langle \vec{X}, \vec{Y}\rangle}} = e(g_1, g_2)^{\alpha_1 s u_k}$. If the device has been revoked, then we cannot calculate it since we need $\langle \vec{X}, \vec{Y}\rangle \neq 0$. We then calculate $E = \prod_{i=2}^{R} E_i^{y_i} = (f_1^{-\langle \vec{X}, \vec{Y}\rangle} \prod_{i=1}^{R} f_i^{y_i})^{\beta}$. Using the above result, we calculate $\xi_1' = \left(\frac{e(E, C_0')}{e(C_0'', D_2)}\right)^{\frac{-1}{\langle \vec{X}, \vec{Y}\rangle}} = e(g_1, g_2)^{\alpha_1 s \beta}$. If the device has been revoked, then we cannot calculate it since we need $\langle \vec{X}, \vec{Y}\rangle \neq 0$. Using the linear reconstruction property of the LSSS access structure, meaning that role credentials are prepared for verification, we calculate $\xi_2 = \prod_{i \in I} \left(e(C_i, D_{\pi(i),0}) \cdot e(K_{\rho(i)}, C_0')\right)^{\omega_i} = \prod_{A_k \in \mathcal{A}} e(g_1, g_2)^{s t_k \delta}$. Finally, we recover the message m by computing $C_0 \cdot \xi_2 \cdot \frac{e(D_{0,\tau_{dec}}, C_0''') \cdot \xi_1'}{e(D_{1,\tau_{dec}}, C_0')} \cdot \prod_{A_k \in \mathcal{A}} \frac{\xi_{k,1}}{e(D_{k,1}, C_{k,0}')} = m$ where the role attributes cancel out with the ones embedded in the access policy (linear reconstruction property), while the time interval fits in the decryption time period (set cover mechanism).

Security Model. To prove MATRaCAE secure, either one role authority whose some attributes are included in the access policy is honest or the time authority is honest. If all authorities are malicious and collude, then the key generation can easily be altered to the advantage of these authorities. W.l.o.g., we assume that there is an honest role authority.

We consider a *selective* security model defined by a game between an adversary $\mathcal{E}$ and a challenger $\mathcal{C}$ (Waters, 2011). $\mathcal{E}$ first *selects* a challenged access structure (M^*, ρ^*), a challenged revocation list $\mathcal{R}^*$, a challenged set $\mathcal{A}^*$ of role authorities whose attributes are in (M^*, ρ^*), and a challenged decryption time period T_{dec}^*. She then receives the public parameters and authorities' public keys. $\mathcal{E}$ can query devices' secret keys that cannot be used to decrypt CT^*. $\mathcal{E}$ *selects* an honest authority $A_{k^*} \in \mathcal{A}^*$ (Chase, 2007). Therefore, she can request secret keys for a device with identity ID and attribute set S_{ID} as long as the device has insufficient attributes from A_{k^*} to decrypt.

Initialization: $\mathcal{E}$ submits (M^*, ρ^*), $\mathcal{R}^*$ and T_{dec}^* to $\mathcal{C}$. She also determines $\mathcal{A}^*$ of role authorities whose attributes are in (M^*, ρ^*) and one honest authority $A_{k^*} \in \mathcal{A}^*$.

Setup: $\mathcal{C}$ runs the algorithms Setup, RAKeyGen and TAKeyGen and gives to $\mathcal{E}$ the public parameters PP, the public keys PK_k for all A_k and the public key PK for B.

Query Phase 1: $\mathcal{E}$ makes secret key queries corresponding to the device with identity ID such that:

- The secret keys $RSK_{ID,k}$ result from the role attribute sets $S_{ID,k}$;
- The secret key TSK_{ID} results from the time range T_{ID}.

At least one of the following conditions must hold:

- Let $S_{ID} = \cup_{A_k \in \mathcal{A}^*} S_{ID,k}$ be the disjoint union of all the role attribute sets $S_{ID,k}$ of the device with identity ID and associated with $A_k \in \mathcal{A}^*$. S_{ID} does not satisfy (M^*, ρ^*), meaning that there must be at least one honest authority $A_{k^*} \in \mathcal{A}^*$ from which $\mathcal{E}$ never requests enough attributes to decrypt CT^*. The honest authority A_{k^*} replies such that S_{ID,k^*} does not satisfy (M^*, ρ^*), meaning that (M^*, ρ^*) cannot contain attributes from A_{k^*} only. In addition, $\mathcal{E}$ never queries the same authority twice with the same identity ID.
- $ID \in \mathcal{R}^*$, meaning that the device has been revoked.
- T_{dec}^* is not completely covered in T_{ID}, meaning that τ_{dec}^* and all its prefixes are not in the set cover $\mathbb{T}$

of T.
Challenge: $\mathcal{E}$ submits two messages m_0 and m_1 of equal length. $\mathcal{C}$ picks a random bit $b \in \{0,1\}$ and encrypts m_b using (M^*, ρ^*), $\mathcal{R}^*$, T^*_{dec} and $A_{k^*} \in \mathcal{A}^*$. The challenged ciphertext CT^* is given to $\mathcal{E}$.
Query Phase 2: This phase is similar to Phase 1.
Guess: $\mathcal{E}$ outputs $b' \in \{0,1\}$ and wins if $b' = b$.

The advantage of $\mathcal{E}$ in the game is defined as $Adv_{\mathcal{E}} = Pr[b' = b] - 1/2$. MATRaCAE is said to be *selectively* secure if no probabilistic polynomial-time $\mathcal{E}$ has non-negligible advantage in the above game.

Security Proof Sketch. We only sketch the tricks used in our security proof due to page limitation. Let a *reduction* be as follows: if one can break MATRaCAE, then one can break the Decisional q-BDHE assumption. Assuming that the Decisional q-BDHE assumption holds, then there is no probabilistic polynomial-time $\mathcal{E}$ that can selectively break MATRaCAE with a challenged access structure (M^*, ρ^*), a challenged revocation list $\mathcal{R}^*$ and a challenged decryption time period T^*_{dec}, along with a set $\mathcal{A}^*$ of role authorities whose attributes are in (M^*, ρ^*) and an honest authority $A_{k^*} \in \mathcal{A}^*$.

When proving our solution secure, we need the reduction to *program* the challenged ciphertext CT^* into the public parameters PP. An attribute may be associated with multiple rows in the challenged matrix M^*, meaning that the function ρ^* is not injective. This is similar to a value appearing in different leaves in a tree. For instance, let $\rho^*(i) = z$ for f_z based on the i-th row of the matrix M^*. If $z = \rho^*(i) = \rho^*(j)$ for some i, j such that $i \neq j$, then this is a problem since we have to program both rows i and j. In the reduction, the above conflict is solved by using different elements in the Decisional q-BDHE assumption. We can thus program different rows of the matrix M^* into one element corresponding to an attribute.

4 EXPERIMENTAL ANALYSIS

While MATRaCAE extends LYZL (Liu et al., 2018), we choose to not compare the two schemes. By design, LYZL is more efficient than MATRaCAE. The participation of multiple authorities (rather than a single one) in MATRaCAE makes key generation an heavier process by design. We showed in Section 2 that our choice of binary trees rather than 31-ary trees is more pertinent and effective for our time-sensitive IoT use cases. Note that only a theoretical analysis was given in (Liu et al., 2018), making the claims about its deployability in realistic environments (e.g. a business) limited.

Environment. We test our solution on a Raspberry Pi 4B with a Quad Core ARM64 Cortex-A72 CPU running at 1.5 GHz with 8 GB of 3200 MHz SDRAM. The Raspberry Pi is a low-cost accessible device and is a realistic assumption of the type of computational power that IoT devices will have in a near future. Indeed, it has been claimed recently that IoT devices' CPU and GPU double every 3-4 years (Sun et al., 2020). The programming language is `Python`3.6 and the library is Python-based Charm Crypto (Akinyele et al., 2011). For all our benchmarks, we execute 1000 tests and calculate the average time (in milliseconds).

Parameters. In IoT, access policies contain up to 30 attributes and devices are allocated around 10 attributes (Ambrosin et al., 2016; Yao et al., 2015). Roles can be related to the IoT functionalities, locations and permissions to specific operations such as Read and Write. Our implementation considers short time intervals. Algorithms RAKeyGen and RUKeyGen can be run in parallel, hence, we only test MATRaCAE with one role authority. Unless specified, we consider the following parameters for our testing: (i) The role authority A_k has 4 attributes; (ii) The device has 2 attributes w.r.t. the role authority A_k; (iii) The time period consists of 16 days ($T = 5$); (iv) The access policy contains 4 attributes (M has 4 rows); (v) Decryption requires 2 attributes (2 rows will be used). We choose to not test bigger numbers of attributes, since we aim for closely matching realistic IoT frameworks (Ambrosin et al., 2016; Yao et al., 2015). Moreover, the efficiency of MATRaCAE will be impacted with large attribute numbers since many components (e.g., keys, ciphertexts) depend on those numbers.

Elliptic Curves. Implementing MATRaCAE requires to generate cyclic groups of prime orders built from an elliptic curve. We selected the following elliptic curves (Miyaji et al., 2000; Galbraith, 2001; Akinyele et al., 2011): (i) SS512 and SS1024 with 512-bit and 1024-bit base fields respectively; (ii) MNT curves with 159-bit, 201-bit and 224-bit base fields respectively. The results are shown in Figure 5. We provide the security level (in bits) in brackets aside the name of the elliptic curve. RAKeyGen and TAKeyGen have similar time results for all elliptic curves. Except with SS1024, Setup and Encrypt also get comparable time outputs. While SS1024 offers the highest security level, the running times of most of the algorithms are noticeably impacted. Decryption with curves MNT159 and MNT201 requires around twice the time with SS512, for a similar security level. While MNT224 guarantees a higher secu-

Curve/Algorithm	Setup	RAKeyGen	TAKeyGen	RUKeyGen	TUKeyGen	Encrypt	Decrypt
SS512 (80)	8.1	4.0	9.4	21.3	31.7	26.8	18.7
SS1024 (112)	92.7	4.3	5.8	240.6	387.9	193.7	300.4
MNT159 (70)	7.5	2.3	3.5	12.9	21.8	21.0	37.8
MNT201 (90)	10.1	3.0	3.9	16.6	27.9	23.5	41.9
MNT224 (100)	13.0	3.8	5.0	21.5	36.7	28.1	61.2

Figure 5: Running times (ms) of all algorithms w.r.t. elliptic curves.

rity level than SS512, the decryption algorithm takes 3 times longer than the former. Based on a trade-off between efficiency and security, we select the curve SS512 for subsequent tests.

Revocation. Let $R-1$ be the maximum number of revoked devices. We are interested in observing how the parameter R affects the execution time of MATRaCAE. We test all algorithms except RAKeyGen and TAKeyGen for which R is not an input. We conduct a first experiment with $R \in \{5, 10, 15, 20, 25, 30\}$. Result are shown in Figure 6. We observe that R has a low impact on encryption and decryption, as expected. Indeed, they rather depend on the value $r = 4$ that is fixed in the experiment. The running time of Setup depends on R, with a light increase with larger values. The running times of RUKeyGen and TUKeyGen are linear in R. Hence, R must remain reasonable to permit an interesting trade-off between the frequency of key updates and the length of the list $\mathcal{R}$. We suggest that R can be up to 15 to keep the running time of RUKeyGen below 100 ms. A larger value would negatively impact the applicability of MATRaCAE by noticeably slowing down device key management in the IoT network. We conduct a second experiment with $R = 10$ (i.e. the maximum number of revoked devices is 9) and $r \in [1, 9]$. $\mathcal{R}$ and r are inputs of Encrypt and Decrypt only. We are interested in seeing how those two algorithms are affected by r. The results are shown in Figure 7. Encryption and decryption timings linearly increase with r. The effect is stronger for encryption than for decryption. Larger the number of revoked devices is, more computing resources are needed for encryption.

Binary tree. We vary the tree depth T in $[5, 12]$. TAKeyGen and TUKeyGen depend on T. The execution time of those algorithms is impacted by the construction of the tree and the execution of the set cover to find the minimum number of nodes to represent the time interval. The results are shown in Table 1. For $T \in [5, 8]$, the time required to build the binary tree and to find the minimum cover set is strictly less than 1 ms. For $T = 9$ and above, the time increases exponentially, since the number of leaves scales with 2^T. In IoT, it is important to keep the time required to

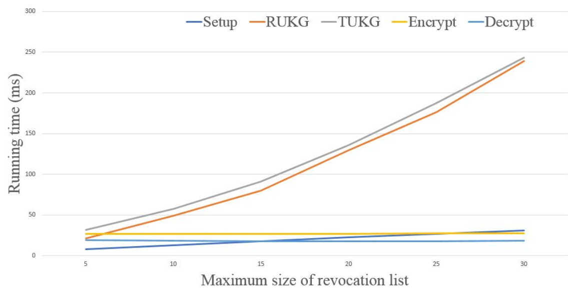

Figure 6: Running times (ms) of Setup, RUKeyGen, TUKeyGen, Encrypt and Decrypt w.r.t. R.

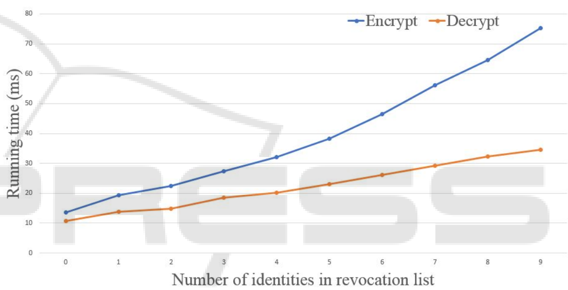

Figure 7: Running times (ms) of Encrypt and Decrypt w.r.t. r.

build the binary tree below 1 ms. For instance, temperature sensors collect data once every few minutes, thus require very short time periods for access (few days), so small trees. Defining trees with reasonable depth moderates storage costs, hence $T = 5$ is a judicious choice.

Key Generation. The number of attributes controlled by a role authority A_k has a linear influence on the time needed to generate the keys PK_k and SK_k. Let this number vary in $[1, 20]$. The results are shown in Figure 8. Each attribute adds around 1 ms in computing the keys. The number of role attributes per device has an impact on the time required to generate the key $RSK_{ID,k}$. Let the number of attributes from A_k be 15 and the number of role attributes per device

Table 1: Running times (ms) of combined TAKeyGen and TUKeyGen w.r.t. T.

T	5	6	7	8	9	10	11	12
Time	< 1	< 1	< 1	< 1	1	2	3	6

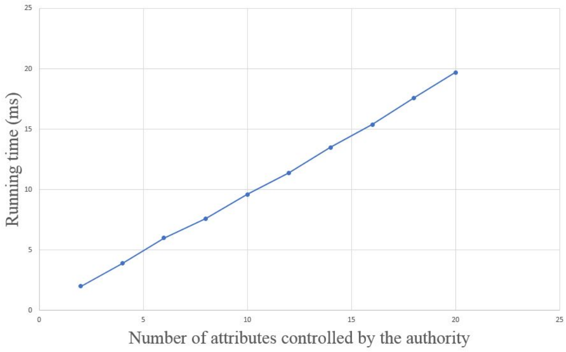

Figure 8: Running times (ms) of RAKeyGen w.r.t. role attributes (authority).

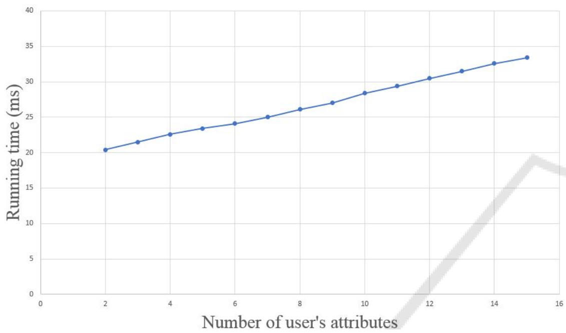

Figure 9: Running times (ms) of RUKeyGen w.r.t. role attributes (device).

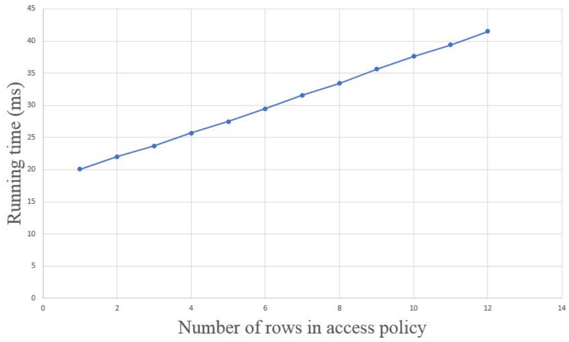

Figure 10: Running times (ms) of Encrypt w.r.t. number of attributes in the access policy (rows).

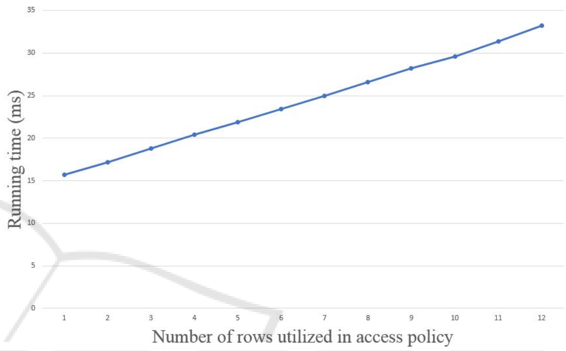

Figure 11: Running times (ms) of Decrypt w.r.t. number of attributes for decryption (rows).

vary in $[2, 15]$. The results are shown in Figure 9. We observe that the time needed to generate $RSK_{ID,k}$ is linear in the number of role attributes given to the device. Each attribute adds around 1 ms in computing the key. By having multiple role authorities, we manage to dispatch all the role attributes among them such that each role authority has only a small attribute subset and shares the computing resources to compute the devices' keys.

Encryption-Decryption. Let the number of attributes in the access policy vary in $[1, 12]$. This number (i.e. the number of rows in the matrix M) has an effect on the running time of Encrypt. The results are shown in Figure 10. The running time of Encrypt is linear in this number. 2 ms are added for each extra attribute. There could be up to 30 attributes in the access policy (Ambrosin et al., 2016; Yao et al., 2015). With 30 attributes, a message could be encrypted in 76 ms, which is reasonable. Decrypt depends on the number of matrix rows needed to recover the message. Let the number of attributes for decryption (i.e. the number of rows) vary in $[1, 12]$. The results are shown in Figure 11. The running time of Decrypt is linear in the number of attributes needed for decryption. 1.5 ms are added for each extra attribute. There could be up to 10 attributes needed to decrypt a ciphertext (Ambrosin et al., 2016; Yao et al., 2015). With 10 attributes, a message could be recovered in 30 ms, which is rational.

5 CONCLUSION

In this paper, we designed a new solution called MATRaCAE for secure access control in IoT, using CP-ABE with attributes representing device roles and time intervals and equipped with a direct revocation mechanism. We devised a novel approach based on binary trees for securing time-sensitive data exchanges in IoT. This allows us to find an interesting, yet effective, trade-off between the frequency of key updates and the length of the revocation list. We gave the intuition to prove MATRaCAE secure under the Decisional BDHE assumption. Implementation and evaluation showed that MATRaCAE is fully deployable in IoT.

REFERENCES

AboDoma, N., Shaaban, E., and Mostafa, A. (2021). Adaptive time-bound access control for internet of things in fog computing architecture. *Int. J. of Comp. and App.*, pages 1–12.

Akinyele, J. A., Green, M. D., and Rubin, A. D. (2011). Charm: A framework for rapidly prototyping cryptosystems. Cryptology ePrint Archive, Report 2011/617.

Ali, Z. H., Ali, H. A., and Badawy, M. M. (2015). Internet of Things (IoT): Definitions, Challenges and Recent Research Directions. *Int. J. of Comp. App.*, 128(1):37–47.

Ambrona, M. and Gay, R. (2023). Multi-authority abe for non-monotonic access structures. In *PKC'23*, pages 306–335.

Ambrosin, M., Anzanpour, A., Conti, M., Dargahi, T., Moosavi, S. R., Rahmani, A. M., and Liljeberg, P. (2016). On the Feasibility of Attribute-Based Encryption on Internet of Things Devices. *IEEE Micro*, 36(6):25–35.

Beimel, A. (1996). *Secure Schemes for Secret Sharing and Key Distribution*. PhD thesis, Israel.

Bethencourt, J., Sahai, A., and Waters, B. (2007). Ciphertext-policy attribute-based encryption. In *SP'07*, pages 321–334.

Chase, M. (2007). Multi-authority attribute based encryption. In *TCC'07*, pages 515–534.

Datta, P., Komargodski, I., and Waters, B. (2021). Decentralized multi-authority abe for dnfs from lwe. In *EUROCRYPT'21*, pages 177–209.

Galbraith, S. D. (2001). Supersingular curves in cryptography. In *ASIACRYPT'01*, pages 495–513.

Guillevic, A. (2013). Comparing the pairing efficiency over composite-order and prime-order elliptic curves. In *ACNS'13*, pages 357–372.

Hwang, Y. H. (2015). IoT Security & Privacy: Threats and Challenges. In *IoTPTS'15*.

Liu, J. K., Yuen, T. H., Zhang, P., and Liang, K. (2018). Time-based direct revocable ciphertext-policy attribute-based encryption with short revocation list. In *ACNS'18*, pages 516–534.

Liu, Z., Wang, F., Chen, K., and Tang, F. (2020). A new user revocable ciphertext-policy attribute-based encryption with ciphertext update. *Secur. Commun. Netw.*, 2020:1–11.

Lu, X., Fu, S., Jiang, C., and Lio, P. (2021). Security and privacy challenges for intelligent internet of things devices view this special issue. *Sec. and Comm. Netw.*, 2021.

Macedo, D., Guedes, L. A., and Silva, I. (2014). A dependability evaluation for internet of things incorporating redundancy aspects. In *ICNSC'14*, pages 417–422.

Mekki, K., Bajic, E., Chaxel, F., and Meyer, F. (2019). A comparative study of lpwan technologies for large-scale iot deployment. *ICT Express*, 5(1):1–7.

Miyaji, A., Nakabayashi, M., and Takano, S. (2000). Characterization of elliptic curve traces under FR-reduction. In *ICISC'00*, pages 90–108.

Oualha, N. and Nguyen, K. T. (2016). Lightweight attribute-based encryption for the internet of things. In *ICCCN'16*, pages 1–6.

Pa, Y. M. P., Suzuki, S., Yoshioka, K., Matsumoto, T., Kasama, T., and Rossow, C. (2015). IoT-POT: Analysing the Rise of IoT Compromises. In *WOOT'15*, pages 9–9.

Patel, M., Shangkuan, J., and Thomas, C. (2017). What's new with the Internet of Things? Technical report, McKinsey.

Pham, C., Lim, Y., and Tan, Y. (2016). Management architecture for heterogeneous iot devices in home network. In *GCCE'16*, pages 1–5.

Rouselakis, Y. and Waters, B. (2015). Efficient statically-secure large-universe multi-authority attribute-based encryption. In *FC'15*, pages 315–332.

Sahai, A., Seyalioglu, H., and Waters, B. (2012). Dynamic credentials and ciphertext delegation for attribute-based encryption. In *CRYPTO'2012*, pages 199–217.

Sahai, A. and Waters, B. (2005). Fuzzy identity-based encryption. In *EUROCRYPT'05*, pages 457–473.

Shamir, A. (1985). Identity-based cryptosystems and signature schemes. In *CRYPTO'84*, pages 47–53.

Sun, Y., Agostini, N. B., Dong, S., and Kaeli, D. (2020). Summarizing CPU and GPU design trends with product data. arXiv 1911.11313.

Waters, B. (2011). Ciphertext-policy attribute-based encryption: An expressive, efficient, and provably secure realization. In *PKC'11*, pages 53–70.

Yan, X., Tu, S., Alasmary, H., and Huang, F. (2023). Multi-authority ciphertext policy-attribute-based encryption (MA-CP-ABE) with revocation and computation outsourcing for resource-constraint devices. *MDPI Appl. Sc.*, 13(20).

Yang, K. and Jia, X. (2014). Expressive, efficient, and revocable data access control for multi-authority cloud storage. *IEEE Trans. on Paral. and Dist. Syst.*, 25(7):1735–1744.

Yao, X., Chen, Z., and Tian, Y. (2015). A lightweight attribute-based encryption scheme for the internet of things. *Future Gener. Comput. Syst.*, 49(C):104–112.

Zhang, J., Li, T., Jiang, Q., and Ma, J. (2022a). Enabling efficient traceable and revocable time-based data sharing in smart city. *Eurasip J. on Wirel. Comm. and Netw.*, 2022(3).

Zhang, Q., Wang, S., Zhang, D., Wang, J., and Zhang, Y. (2019). Time and Attribute Based Dual Access Control and Data Integrity Verifiable Scheme in Cloud Computing Applications. *IEEE Access*, 7:137594–137607.

Zhang, R., Li, J., Lu, Y., Han, J., and Zhang, Y. (2022b). Key escrow-free attribute based encryption with user revocation. *Inf. Sc.*, 600:59–72.

Zhang, T., Wang, C., and Chandrasena, M. I. U. (2023). Blockchain-assisted data sharing supports deduplication for cloud storage. *Connect. Sc.*, 35(1).

Zhang, Y., Nakanishi, R., Sasabe, M., and Kasahara, S. (2021). Combining IOTA and Attribute-Based Encryption for Access Control in the Internet of Things. *MDPI Sensors*, 21(15):50–53.