# *SAFE*: Secure and Flexible Encryption for Dynamic Team Communications in Disaster Management

Hongmiao Yu*, Jiachen Chen†, K. K. Ramakrishnan*
* Department of Computer Science and Engineering, University of California Riverside, Riverside, USA
† WINLAB, Rutgers University, North Brunswick, NJ, USA.
hyu125@ucr.edu, jiachen@winlab.rutgers.edu, kk@cs.ucr.edu

2023 IEEE 31st International Conference on Network Protocols (ICNP) | 979-8-3503-0322-3/23/$31.00 ©2023 IEEE | DOI: 10.1109/ICNP59255.2023.10355587

***Abstract*—Name-based publish/subscribe systems using Information-Centric Networking (ICN) principles can provide a flexible and efficient framework for communication in disaster situations. Efficient, secure dissemination of information can play a critical role in disaster management. But, secure and authenticated group communications that maintain confidentiality and integrity remain a challenge.**

**In this paper, we design a flexible and efficient encryption framework *SAFE* that leverages graph-based naming frameworks for providing role-based communication among first responders. We study the suitability of message-oriented encryption where the sender leverages the name hierarchy, and compare it with a key-oriented encryption scheme that requires the receiver to utilize appropriate keys to decrypt based on the publisher-targeted name for the message. Both encryption schemas can be built with attribute-based encryption (ABE) or public key encryption (PKE) implementations. We find message-oriented encryption provides the needed flexibility for dynamic environments when communicating with members changes frequently. With message-oriented encryption, we further address key revocation and support for infrastructure-less environments in disaster situations and consider the tradeoff between flexibility and optimization for large relatively static communication groups.**

**We evaluate both encryption schemas built on top of ABE and PKE. We examine the key generation time, ciphertext length, encryption, and decryption time, and see that *SAFE*'s design is the most suitable for large and dynamically changing groups.**

## I. Introduction

Effective communication among first responders in a disaster environment is critical to the success of public safety missions. The key to effective communication during a disaster includes communicating at different levels of granularity in an evolving incident command chain. Members from different departments or teams may need to receive different messages based on their role and position in the organization or incident management. Additionally, first responders may frequently move between different teams with different objectives. A framework that flexibly allows for dynamic changes to the communication framework and enables first responders to communicate efficiently is highly desirable. Having a well-defined naming framework, typical of ICN approaches [1], [2], can accommodate the needs of emergency communication. Information-Centric Networking (ICN)-based disaster communication systems [2]–[4] provide the basic framework for flexible and efficient name-based communication to meet this requirement for emergency communication. Publish/Subscribe (pub/sub) systems (*e.g.,* [5], [6]) are convenient for information dissemination, and provide the necessary push-based information delivery needed for one-to-many first-responder communication that is typically needed in our context [7], [8].

979-8-3503-0322-3/23/$31.00 ©2023 IEEE

Compared to efficiency, secure communication is even more critical to first responders who need to exchange information when responding to disasters. Leaking confidential information could be harmful whether it falls into the hands of malicious individuals, or even the general public as it may cause unnecessary panic. In the spirit of ICN [1], [9], we seek to secure the data that is transmitted rather than the communication channel. We use encryption of the transmission between first responders to maintain confidentiality. A variety of encryption schemes, such as public key encryption (PKE) [10], [11], identity-based encryption (IBE) [12], or attribute-based encryption (ABE) [13] could be utilized. However, there is a need to augment these schemes to meet the needs of emergency communication, to be able to flexibly identify the recipients for a message. The challenge is to be able to specify the recipient identities (roles identified in the namespace) and their authorization to decrypt the content for each message. Existing methods, including IP-multicast, and ICN (where the content is identified in a request/response, such as in NDN [1], [2]) fail to adequately address this issue.

To ensure messages are exchanged securely for communicating among members of a dynamically formed group in such a role-based namespace, we seek a method for encrypting messages that supports the namespace hierarchy. A message sent to a group identified by a destination name should only be decryptable by subscribers of that name and its descendants in the hierarchy. Traditional encryption schemes fail to take aspects of groups such as the namespace hierarchy into consideration and are not flexible in accommodating dynamic groups (*e.g.,* dispatching units for specific roles, day/night shift). For PKE, public and secret keys have to be generated together as pairs. If we generate a different public-secret key pair for each subscriber, it will be difficult and extremely inconvenient for publishers to track which subscribers belong to which name. For IBE and ABE, the sender (encryptor) and receiver (decryptor) could encrypt or decrypt based on identities or a branch of attributes related to names. However, the identities and attributes themselves do not have an associated hierarchy. Previous work has utilized ABE to secure constant topic-based names ICN by encrypting messages based on a combination of several names with similar properties [14]. However, this approach does not meet the requirements of a role-based name system in a disaster environment, where names can dynamically change. The use of name combinations would impose limitations on the flexibility of the system. First responders may frequently have to operate in disconnected/infrastructure-less environments, where they may not be able to acquire new

Authorized licensed use limited to: Univ of Calif Riverside. Downloaded on October 12,2024 at 15:43:21 UTC from IEEE Xplore. Restrictions apply.

keys. It is imperative that they still be able to communicate securely in such situations (*e.g.,* in a shelter). Therefore, an encryption mechanism beyond one of these existing mechanisms of PKE, ABE, or IBE is needed.

In this paper, we design *SAFE*, an authenticated, confidential message exchange framework using encryption techniques that leverage the graph-based namespace used for communication among first responders during disaster management [7].

For encrypted information exchange, using any of the techniques we study here, ABE (either Key Policy Attribute Based Encryption (KP-ABE) [15] or Ciphertext-Policy Attribute-Based Encryption (CP-ABE) [16]) or PKE [10], [11], a publisher sends a message with a name or set of names (name set) embedded in the ciphertext. That ciphertext can be decrypted by the subscribers of the target name or of its descendants of the name in its sub-graph based on the name set in the key(s) that they have. We examine the distinct approaches for how the name set (or attribute set) is included in the message or in the key. A *message-oriented approach* would include a name set (or attributes) in each message and the key at the receiver requires only a single name that finds a match in the name set. In a *key-oriented approach*, the message a publisher encrypts would include a single name. The key of a receiving subscriber embeds a set of names, including those of its ancestors. The underlying key generation requires a set of names in each key.

We generalize these message and key-oriented encryptions further by having a set of names ($\mathbb{N}_k$) in each key, and correspondingly, a set of names ($\mathbb{N}_m$) in each published message. We evaluate both the message and key-oriented approaches, using a range of underlying encryption approaches (PKE, Key Policy, Ciphertext Policy) both qualitatively and for performance (CPU consumption and network bandwidth).

We see in our evaluations that a message-oriented interface is superior in terms of flexible processing and bandwidth efficiency. Note that with message-oriented encryption, other fundamental encryption methods can be implemented. In this paper, we build the message-oriented interface on top of three different fundamental encryption implementations: KP-ABE, CP-ABE, and multi-envelope public key encryption (ME-PKE). With both KP-ABE and CP-ABE, we use a unique ID related to the name [2] as an attribute and manage the attributes to fulfill the requirements for the message-oriented solution. The ME-PKE we propose utilizes public key encryption to deliver messages to multiple receivers. We compare each implementation in terms of key generation, communication overhead, encryption performance, and decryption performance. As we show, *SAFE*, our message-oriented KP-ABE scheme, has the best encryption performance, the least communication overhead, and the second-best decryption performance.

We briefly summarize the contributions of this paper:

- We build an abstraction framework utilizing the well-known encryption schemes (ABE, PKE) to provide an encrypted information exchange integrated into a graph-based naming framework for information-centric pub/sub communication.
- We design, develop, and evaluate *SAFE*, a message-oriented and key-oriented encryption solution for confidential communication between publishers and subscribers.
- We compare the key generation, communication overhead, encryption, and decryption performance of several encryption implementations: MultiEnvelope-PKE (ME-PKE), KP-ABE, and CP-ABE using a message-oriented abstraction.
- We further enhance the base message-oriented KP-ABE solution, with mechanisms that support richer semantics and key revocation for a large group of publishers/subscribers that use a name-based communication infrastructure.
- *SAFE* facilitates flexible, per-message dynamic group information exchange even within disconnected and infrastructure-less environments.

## II. Background & Related Work

### A. Name-based Publish/Subscribe

First responders need to communicate with different groups of people, depending on the situation and the information they need to communicate. Despite its adoption and deployment in many situations, IP multicast has limitations since IP multicast group addresses do not capture the semantic relationships within and between groups (*e.g.,* hierarchical groups). Name-based multicast, leveraging the concept of ICN [17], which shows a number of benefits in disaster environment [18], can overcome a number of these limitations, and provides the appropriate level of flexibility, dynamics and structure among multicast groups [6]–[8], [19].

The ICN-based publish/subscribe model [6] has been proposed for its efficient group multicast, demonstrating its benefits not only in disaster scenarios but also in other contexts such as IoT [20] and smart cities [21]. This approach overcomes the scalability and complexity issues of traditional long-lived connection-based [22] or broker-based [23] publish/subscribe systems. CNS [8] further improves the publish/subscribe model by introducing a role-based namespace, which facilitates the dissemination of information. POISE [7] extends CNS by relieving the strict hierarchy restrictions and to a graph-based dynamic namespace.

A graph-based dynamic namespace allows first responders from different organizations and incident management hierarchies to use publish/subscribe to communicate freely. First responders subscribe to names based on their role. A message sent to a particular name will result in all subscribers of names under that name in the hierarchy to also receive the message [7]. This ensures that useful information sent to an organization is disseminated to the appropriate groups within the organization. This is particularly important when a dynamic team is formed for incident management.

A sub-namespace could be added for an incident, such as a particular city's firefighting team in its entirety. An incident topic may be created, and different departments may be added to the incident management by attaching their namespace hierarchy by a dispatcher or incident commander. For example, when a fire occurs, commanders could create a "Fire Incident" and connect all relevant teams to the incident.

Assume that we have a namespace shown in Fig. 1. The yellow nodes are part of the organizational namespaces (departments and units), while the blue nodes are the incident-



Authorized licensed use limited to: Univ of Calif Riverside. Downloaded on October 12,2024 at 15:43:21 UTC from IEEE Xplore. Restrictions apply.

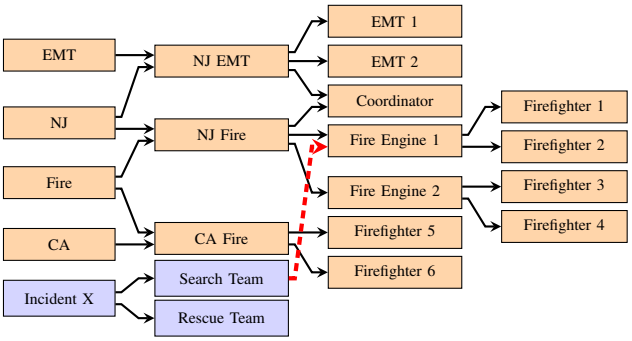



Fig. 1: Example Namespace for Disaster Management

related names (roles). Say for example the dispatcher is assigning units for "Search Team" as part of "Incident X". (*e.g.,* dispatching "Fire Engine 1" (including "Firefighter 1" and "Firefighter 2") as part of the search team (shown as a red dashed line in the figure)). This allows for the namespace to be dynamically managed as the incident evolves. Communication among members of various groups often needs to be kept confidential. This is achieved through proper encryption.

### B. Attribute-based Encryption

Traditional PKE (*e.g.,* RSA [10]) requires the encryptor to get the public key certificate in order to generate a secret message. IBE [12] allows the public key to be an arbitrary string (*e.g.,* the email address of the receiver). ABE [13], [15], [16] further extends IBE by allowing the public key string to not be atomic, but be a set of attributes (*e.g.,* address, roles). To decrypt a ciphertext, the decryptor has to hold a key that matches the attributes used when the message is encrypted. Based on how the attributes are matched, ABE can be generally classified into 2 categories: KP-ABE [15] and CP-ABE [16].

**KP-ABE:** KP-ABE allows the ciphertext to embed a set of attributes ($\gamma_m$, *e.g.,* "$attr_1, attr_2, \ldots$") and the key to encode an access policy ($\mathbb{A}_k$). An access policy is a conjunction of attributes with relationships ("and" [$\wedge$], "or" [$\vee$]), *e.g.,* "$(attr_1 \vee attr_2) \wedge attr_3$". The key can decrypt the message only if $\gamma_m$ satisfies $\mathbb{A}_k$. For example, the key with access structure "$(attr_1 \vee attr_2) \wedge attr_3$" can decrypt message with attributes "$attr1, attr3$", but not "$attr1, attr2$".

**CP-ABE:** Similar to KP-ABE, CP-ABE also encrypts and decrypts messages based on attributes. However, to allow senders to have better control of the recipients of the message, CP-ABE embeds the *policy* ($\mathbb{A}_m$) in the message instead of the keys. Keys in CP-ABE only contain a set of attributes ($\gamma_k$). A key can decrypt a message only when $\gamma_k$ satisfies the policy $\mathbb{A}_m$ in the ciphertext. *E.g.,* a receiver with key "$attr_1,\ attr_2$" can decrypt messages with policies "$attr_1 \wedge (attr_2 \vee attr_3)$" and "$attr_1 \vee (attr_2 \wedge attr_3)$" but not "$attr_1 \wedge attr_2 \wedge attr_3$".

### C. Secure Group-Communication Solutions

One approach to address group communication is multicast encryption. By allowing all recipients to share the same decryption key, multiple receivers can decrypt messages sent by a sender. However, this approach encounters the challenge of efficient key revocation [24], as revoking the key of even an individual recipient requires re-issuing a key to all the others.

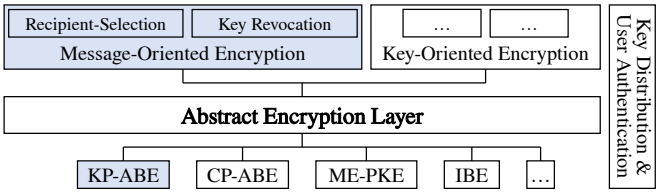



Fig. 2: *SAFE* Design Overview

It also suffers from flexibility issues. The sender can only choose to send to some predefined (usually coarse-grained) groups instead of being able to decide the proper recipient on a message-to-message basis.

A current solution for communication among first responders during disasters, FirstNet [25] utilizes unicast secure channels for each receiver, which results in excessive overhead, and is not suitable for disconnected environments.

## III. *SAFE* Design Overview

**Adversary Model:** An adversary may try to recover the plaintext from the ciphertext obtained by sniffing the communication between the publisher and the subscriber, without a key, which our system needs to prevent. Another possible attack by an adversary is to obtain private keys during the key distribution phase. *SAFE* needs to keep the keys secure during the key distribution, allowing access only for the key issuer and subscribers. Another scenario is when a first responder loses his or her subscribed device, *SAFE* must have a key revocation method for rapid removal of those devices no longer trusted. A proper authentication mechanism is needed to prevent unauthorized users from subscribing and publishing. The system should also support perfect forward secrecy [26], [27] and backward secrecy [28], which implies that the use of a key should be time-limited, and the current key can not be used to recover a previous or a future key.

**Design Summary:** The name-based pub/sub message exchange utilizes *SAFE* to achieve confidentiality, authenticate users, and ensure message integrity suitable for communication among first responder groups. We support either a message-oriented or key-oriented encryption mechanism. *SAFE* is built on top of an "abstract encryption layer" that provides basic functions needed for dynamic and efficient encryption. Figure 2 shows the overview of *SAFE*. Subscribers of a name or its descendants would receive a message sent to that name in a name-based pub/sub framework (as in [7]). To provide confidentiality, we introduce an "abstract encryption layer" (AEL) on top of a basic encryption scheme. This AEL serves as an interface to the underlying encryption schemes such as PKE, KP-ABE, and CP-ABE. The AEL manages the abstraction of basic encryption functions such as key generation, encryption, and decryption. We discuss the AEL in more detail in §IV.

The AEL supports two possible approaches, a message-oriented encryption approach and a key-oriented approach. The message-oriented encryption leverages the namespace during the encryption process and embeds the name and its descendants into the encrypted message. On the other hand, the key-oriented solution embeds the name hierarchy in the key



Authorized licensed use limited to: Univ of Calif Riverside. Downloaded on October 12,2024 at 15:43:21 UTC from IEEE Xplore. Restrictions apply.

itself, allowing subscribers to decrypt messages with names that match the names included in the key. We detail these two solutions and outline the advantages and disadvantages of each approach in §V. Message-oriented encryption offers publishers the ability to adapt to namespace changes by just including or excluding names from the updated namespace. Because of this adaptability, we select message-oriented encryption for *SAFE*.

The AEL can be implemented using various base encryption schemes, such as KP-ABE, CP-ABE, ME-PKE, IBE, *etc.* In this paper, we have implemented the AEL using KP-ABE, CP-ABE, and ME-PKE. For KP-ABE and CP-ABE, we utilize names as attributes and embed the name hierarchy in the form of a set of attributes or an access structure. For ME-PKE, we generate distinct public/secret key pairs for each name. We then embed the name hierarchy by encrypting messages with multiple corresponding public keys and letting subscribers hold multiple secret keys. We explain this in §VI.

We propose mechanisms for key distribution and user authentication. It is crucial to encrypt the messages during the key-distribution process to prevent eavesdroppers from obtaining keys. Additionally, we require users to authenticate themselves when subscribing to names, and transmitting messages. This ensures that only authorized users can access the system and guarantees the integrity of the messages (§VII).

On top of the message-oriented encryption, we design rich recipient-selection semantics to provide more flexibility for the publishers to decide who can decrypt each message (§VIII-A). To improve the efficiency in large groups whose members do not change frequently, we also design a solution to achieve better efficiency in terms of message size and encryption/decryption time (§VIII-B).

In a dynamically changing environment, it is critical to enable revocation of the ability of recipients to decrypt messages sent to a role that they are no longer in, and have unsubscribed to the role (*e.g.,* when they are assigned to a different role, or when a "shift" changes). For the message-oriented encryption mechanism, we have developed key revocation. This includes a timeout-based key revocation method and a blacklist-based key revocation method. Implementing these methods can be challenging for any of the base encryption schemes. In the case of timeout-based key revocation, we achieve it by adding a timestamp in each attribute, enabling the control of key usage within a certain time interval. Similarly, for blacklist-based key revocation, we utilize temporary names to effectively blacklist specific subscribers from a group of subscribers who share the same key (§IX).

*SAFE* establishes a relationship between the set of role-based recipient names and the encryption mechanisms to provide an efficient, but flexible, framework. By integrating role-based names, *SAFE* can be incorporated into an IP-based architecture or in existing ICN architectures. The role-based names could be directly implemented in MobilityFirst utilizing Globally Unique IDs (GUIDs). With gateways, the role-based names can also be applied to NDN [1] and DONA [29]. As demonstrated in [30], the role-based name works with IP-based architectures at the application layer, with information dissemination being achieved using overlays.

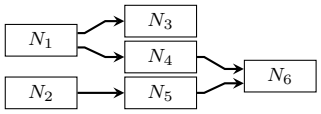



Fig. 3: Example Graph-based Namespace for Encryption

TABLE I: Message-Oriented *vs.* Key-Oriented Solution

| | **Message-Oriented** | **Key-Oriented** |
|---|---|---|
| **Subscribe:** | $N_1 \mapsto N_1$ | $N_1 \mapsto N_1$ |
| (subscribed | $N_2 \mapsto N_2$ | $N_2 \mapsto N_2$ |
| name $\mapsto$ | $N_3 \mapsto N_3$ | $N_3 \mapsto N_1, N_3$ |
| names that | $N_4 \mapsto N_4$ | $N_4 \mapsto N_1, N_4$ |
| subscriber can | $N_5 \mapsto N_5$ | $N_5 \mapsto N_2, N_5$ |
| decrypt ($\mathbb{N}_k$)) | $N_6 \mapsto N_6$ | $N_6 \mapsto N_1, N_2, N_5, N_6$ |
| **Publish:** | $N_1 \mapsto N_1,\ N_3,\ N_4,\ N_6$ | $N_1 \mapsto N_1$ |
| (destination | $N_2 \mapsto N_2,\ N_5,\ N_6$ | $N_2 \mapsto N_2$ |
| $\mapsto$ | $N_3 \mapsto N_3$ | $N_3 \mapsto N_3$ |
| names that | $N_4 \mapsto N_4,\ N_6$ | $N_4 \mapsto N_4$ |
| publisher need | $N_5 \mapsto N_5,\ N_6$ | $N_5 \mapsto N_5$ |
| to encrypt ($\mathbb{N}_m$)) | $N_6 \mapsto N_6$ | $N_6 \mapsto N_6$ |

## IV. Abstract Encryption Layer

We now describe the AEL to abstract the service offered by the underlying encryption scheme and handle the embedding of the name hierarchy with the secure message exchange through encryption. The AEL abstracts 3 algorithms:

**Key Generation ($\mathbb{N}_k$ | SK)**: The key issuer takes a set of names $\mathbb{N}_k$ as input and generates a secret key (SK). $\mathbb{N}_k$ indicates the names SK can decrypt.

**Encrypt ($M$, $\mathbb{N}_m$ | CT)**: The encryptor (publisher) takes message ($M$), a set of names $\mathbb{N}_m$ as input and generates the ciphertext (CT). The name set $\mathbb{N}_m$ indicates the recipients the message is addressed to.

**Decrypt (CT, SK | $M$?)**: The decryptor (subscriber) uses the SK to decrypt the message. The message will be successfully decrypted if $\mathbb{N}_k \cap \mathbb{N}_m$ is not empty.

## V. Encryption in Graph-based Pub/Sub

To discuss the encryption on top of a simple graph-based namespace, we create an example namespace in Fig. 3, a directed acyclic graph structure. We need to allow (or deny) receivers to decrypt the messages based on the semantics of our namespace. All subscribers of the name and its descendants should be able to decrypt the message.

### A. Message-oriented Encryption

The Message-oriented encryption solution embeds the DAG name relationships when encrypting the messages.

The key generation algorithm generates the secret key (SK) for a subscriber of a name $N$, for which the key issuer only needs to take $\mathbb{N}_k$:$\{N\}$ as input. During message-oriented encryption, the publisher uses $\mathbb{N}_m$:{target name and also *all its descendants*} as input. For decryption, all the (authenticated) subscribers of the name in the hierarchy that satisfy $\mathbb{N}_k \cap \mathbb{N}_m$ can decrypt the message.

The left column of Table I shows the details of the message-oriented solution. For example, the subscriber of $N_6$ will only hold the key(s) that could decrypt the message sent to $N_6$. When a publisher send message to $N_1$, it will encrypt with $\mathbb{N}_m$:$\{N_1, N_3, N_4, N_6\}$. Subscribers of $N_6$ will be able to



Authorized licensed use limited to: Univ of Calif Riverside. Downloaded on October 12,2024 at 15:43:21 UTC from IEEE Xplore. Restrictions apply.

decrypt the message by the SK with $\mathbb{N}_k$:$\{N_6\}$ provided to it by the key issuer.

### B. Key-oriented Encryption

An alternative solution is key-oriented encryption, where we embed the hierarchical relationships in the keys. Here, the key generation algorithm takes $\mathbb{N}_k$:{the name itself and *all its ancestors*} as input to generate a key to be issued to subscribers of that name. For the encryption algorithm, the input would be $\mathbb{N}_m$:{the intended recipients' name}. For decryption, the subscriber uses its own key provided by the key issuer that matches its name or that of its ancestors.

For example in the example graph-based namespace, when subscribing to $N_6$, the user should be able to receive messages that are sent to $N_1$, $N_2$, $N_4$, $N_5$ and $N_6$. Therefore, the key issuer will issue the key(s) that with $\mathbb{N}_k$:$\{N_1, N_2, N_4, N_5, N_6\}$. A publisher sending a message to $N_1$ only needs to encrypt the message with $\mathbb{N}_m$:$\{N_1\}$. Subscribers of $N_6$ would use the corresponding key to decrypt the message. The right column of Table I details the decryption key(s) held by a subscriber (and the publisher's action).

### C. Comparing Message-oriented vs. Key-oriented encryption

A significant shortcoming of key-oriented encryption is that the key issuer (the name-certification service, NCS) has to generate new keys for affected subscribers when the namespace is updated.

For message-oriented encryption, the computational overhead may be high for a sender. But this overhead is distributed across all senders and only occurs when sending a message, which is preferable compared to the overhead on a single entity (NCS) with key-oriented encryption. A second concern with message-oriented encryption is that the namespace at some senders may be stale in a disconnected environment. However, this is acceptable as the system behavior matches a first responder's understanding of the namespace.

## VI. Implementation of Encryption Schemes

This section describes the different encryption schemes that can be used to implement the service provided by the AEL. We use KP-ABE, CP-ABE, and PKE as examples to illustrate how the AEL can be implemented. However, other encryption schemes (*e.g.,* IBE) can also be used as the base scheme.

Using Fig. 3, we look at the attributes/keys required for each name from the perspectives of both the publisher and the subscriber in Table II when publishers send messages to $N_1$ and subscribers of $N_6$ decrypt those messages. The name list used with encryption $\mathbb{N}_m$ would contain $N_1$ and its descendants, $N_3, N_4, N_6$ For decryption, the name list $\mathbb{N}_k$, would contains $N_6$ and its ancestors, $N_1, N_2, N_4, N_5$

### A. Key Policy Attribute Based Encryption (KP-ABE)

As indicated in Sec. II, in KP-ABE, the four fundamental primitive algorithms of Setup, Key Generation, Encrypt, and Decrypt provide the functionality required by the AEL. With Setup, the NCS generates the pubic parameters (PK) and a master key (MK, private to the key issuer). Publishers Encrypt the message using PK with a set of attributes ($\gamma_m$) embedded in the message and generate the ciphertext. With Key Generation, the NCS generates a private key for the decryptor with an access structure ($\mathbb{A}_k$) embedded in it. Using the Decrypt algorithm, if the $\gamma_m$ in the ciphertext satisfies the $\mathbb{A}_k$ in the private key, the decryptor could decrypt the message.

Each attribute is a single name, and we utilize a GUID [2] of each name as the attribute. For $\mathbb{A}_k$, we exclusively use the "OR" operator ("$\vee$") in the relationship of the attributes. Thus, when an attribute is present both in $\gamma_m$ and $\mathbb{A}_k$, $\mathbb{A}_k$ will be satisfied, allowing the ciphertext to be decrypted.

The $\mathbb{A}_k$ embedded in the private key comprises all the attributes of names in the set $\mathbb{N}_k$. The attributes of names in $\mathbb{N}_m$ as $\gamma_m$ are embedded in the ciphertext. The ciphertext will be decrypted if $\gamma_m$ and $\mathbb{A}_k$ contain the same attribute. This process naturally corresponds to the operation $\mathbb{N}_k \cap \mathbb{N}_m$, satisfying the requirement of the decryption algorithm in AEL.

Table II is an example of how $\gamma_m$ and $\mathbb{A}_k$ contain names in $\mathbb{N}_m$ and $\mathbb{N}_k$. Since the attribute and name have a one-to-one correspondence, we use the name ($N_x$) to directly indicate the attribute of that name ($attr_{N_x}$). When encrypting the message sent to $N_1$, the publisher uses all the names in $\mathbb{N}_m$, including $N_1, N_3, N_4, N_6$ to compose the set $\gamma_m$ and embed $\gamma_m$ in the ciphertext before sending to $N_1$. When decrypting the message, the private key held by subscribers of $N_6$ will contain $\mathbb{A}_k$ comprising all the names in $\mathbb{N}_k$ ($N_1, N_2, N_4, N_5, N_6$) connected by the "OR" operator ("$\vee$"). $\mathbb{A}_k$ will be $N_1 \vee N_2 \vee N_4 \vee N_5 \vee N_6$. Therefore, $\gamma_m$ finds a match in $\mathbb{A}_k$. Subscribers of $N_6$ can decrypt the message.

### B. Ciphertext Policy Attribute Based Encryption (CP-ABE)

The CP-ABE also has a similar set of four primitives as with KP-ABE. However, the access structure ($\mathbb{A}_m$) is embedded in the **ciphertext**, while the set of attributes ($\gamma_k$) is contained in the **private key**. We also utilize a GUID of each name as an attribute. For $\mathbb{A}_m$, we will also exclusively use the "or" operator ("$\vee$") for the relationship between attributes.

In CP-ABE, the attributes $\gamma_k$ of a name in $\mathbb{N}_k$ are embedded in the private key. $\mathbb{A}_m$ embedded in the ciphertext comprises all the attributes of names in the set $\mathbb{N}_m$. Similar to KP-ABE, the decryption process fulfills the requirement of the decryption algorithm of AEL.

Table II shows the example of how $\mathbb{A}_m$ and $\gamma_k$ contains names in $\mathbb{N}_m$ and $\mathbb{N}_k$ in CP-ABE. When encrypting the message sent to $N_1$, $\mathbb{A}_m$ will be composed of attributes of names in $\mathbb{N}_m$, as $N_1 \vee N_3 \vee N_4 \vee N_6$. During decryption, all the attributes of names in $\mathbb{N}_k$, including $N_1, N_2, N_4, N_5, N_6$, will compose $\gamma_k$ and are embedded in the key held by subscribers of $N_6$. $\gamma_k$ finds a match in $\mathbb{A}_m$ and the message could be decrypted by subscribers of $N_6$.

### C. Multi-Envelope Public Key Encryption

In Public Key Encryption (PKE), keys are generated as public/secret key pairs. The public key (PK) is used to encrypt the message, and the corresponding Secret Key (SK) for decrypting the message.

Using PKE to implement the AEL, the NCS generates a public/secret key pair for each name. Thus, each name is



Authorized licensed use limited to: Univ of Calif Riverside. Downloaded on October 12,2024 at 15:43:21 UTC from IEEE Xplore. Restrictions apply.

TABLE II: Implementation of AEL using Different Encryption Mechanisms (publisher sending to $N_1$, subscriber is $N_6$)

| | KP-ABE | CP-ABE | ME-PKE |
|---|---|---|---|
| **Encrypt** $\mathbb{N}_m$=$\{N_1, N_3, N_4, N_6\}$ | $\gamma_m$: $N_1,\ N_3,\ N_4,\ N_6$ | $\mathbb{A}_m$: $N_1 \vee N_3 \vee N_4 \vee N_6$ | $PK_{N_1},\ PK_{N_3},\ PK_{N_4},\ PK_{N_6}$ |
| **Decrypt** $\mathbb{N}_k$=$\{N_1, N_2, N_4, N_5, N_6\}$ | $\mathbb{A}_k$: $N_1 \vee N_2 \vee N_4 \vee N_5 \vee N_6$ | $\gamma_k$: $N_1,\ N_2,\ N_4,\ N_5,\ N_6$ | $SK_{N_1}, SK_{N_2}, SK_{N_4}, SK_{N_5}, SK_{N_6}$ |

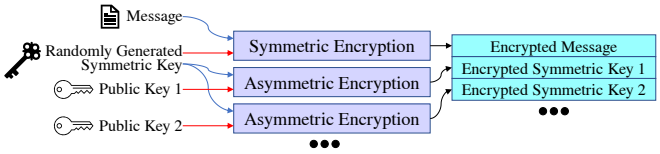



Fig. 4: Process for Multi-Envelope Public Key Encryption

associated with a key pair. To encrypt a message, the publisher needs to use the distinct PKs of names in $\mathbb{N}_m$ to encrypt the message. For decryption, subscribers hold multiple SKs of names in $\mathbb{N}_k$ and try each to see if that SK can correctly decrypt the ciphertext. Therefore, the requirement of AEL is satisfied when $\mathbb{N}_k \cap \mathbb{N}_m$ is not empty, and the receiver can decrypt the message. But to send the message to multiple receivers, PKE requires encrypting the same message with multiple different PKs, which can be expensive. To address this, we proposed a ME-PKE scheme here.

To minimize the computational cost and shorten the length of the ciphertext, we utilize an "envelope structure" (Hybrid Encryption), as in OpenSSL [31] for this multiple public key encryption scenario. The envelope structure is a public key encryption strategy that involves encrypting the message with a randomly generated symmetric key and then using the public key to encrypt this symmetric key instead of the entire message [32]. In our proposed use of the envelope also the publisher generates a random symmetric key and uses it to encrypt the message. However, the symmetric key is then encrypted multiple times with different PKs, as shown in Fig. 4, thus resulting in multiple envelopes. When a subscriber receives the message, it decrypts the symmetric key using its SKs to decrypt the envelope to successfully extract the symmetric key. It then decrypts the message content using the decrypted symmetric key.

Table II gives an example of how a publisher and the subscribers use PKs and SKs. When encrypting a message sent to name $N_1$, since name set $\mathbb{N}_m$ contains $N_1,\ N_3,\ N_4,\ N_6$, the publisher needs to use all the PKs of the names in $\mathbb{N}_m$, which are $PK_{N_1},\ PK_{N_3},\ PK_{N_4},\ PK_{N_6}$, to encrypt the message. For decryption by subscriber(s) of $N_6$, since name set $\mathbb{N}_k$ contains $N_1, N_2, N_4, N_5, N_6$, the subscriber(s) needs to hold $SK_{N_1},\ SK_{N_2},\ SK_{N_4},\ SK_{N_5},\ SK_{N_6}$.The ciphertext could be then decrypted since the corresponding SKs are held by the subscriber.

We provide a detailed quantitative evaluation of KP-ABE, CP-ABE and ME-PKE in Sec.XI, with a particular focus on message-oriented encryption.

## VII. Key Distribution and Authentication

**Key Distribution:** In order to securely distribute private keys of names to its subscribers, a number of measures are needed. First, all key distribution messages need to be encrypted to prevent unauthorized access. To require a private key(s) of a name from the NCS, the subscriber needs to generate its own public/private key pair and share the public key with NCS. Then, the NCS encrypts the required private key(s) for the name/role that the user subscribes to, using that subscriber's public key, and sends those back.

**Subscriber (First Responder) Authentication:** In *SAFE*, subscribers are authenticated prior to requesting keys from the NCS. To achieve this, we can use one of a number of conventional approaches, *e.g.,* a two-step authentication. The first step involves verifying the identity of the first responder, which is carried out offline, potentially with the organization initially physically verifying the identity of the first responders and assign them an authorized device. Alternatively, first responders could authenticate themselves with biometrics of their own devices. The second step involves the NCS verifying the subscription action online. This can be accomplished through various methods, such as two-factor authentication so that the NCS can then issue keys to authenticated subscribers.

**Message Authentication & Integrity:** Attackers could act as publishers and generate false or meaningless messages to subscribers. Also, an attacker could generate a man-in-the-middle attack and modify the original message. Therefore, authenticating a publisher and maintaining message integrity is critical. Publishers can utilize an encrypt-then-MAC mechanism by signing their message using their private key after encrypting the message. Subscribers can then verify the message based on a chain of trust certification. This mechanism is also commonly used in Name Data Networking (NDN) [9]. In our case, the public key of first responders should be signed by their respective departments. The department's public key, in turn, should be signed by a higher-level department, and so on. When first responders publish messages, they include the signed public key along with the message. As a result, subscribers only need to cache a higher-level public key they trust and can verify the signature of its descendants. This ensures that messages are securely transmitted and originate from authorized senders.

## VIII. Optimizations

### A. Supporting Richer Semantics

At the security layer, we can provide richer semantics to have better control of access to the messages, while the network focuses on efficient delivery with simpler graph-based naming for determining senders and receivers. Since the receiver's computation is performed using message-oriented encryption at the AEL, we can support richer semantics than defined in the graph-based namespace at the network layer.

We define an expressive language, starting from basic set operators of "$intersect$" [$\cap$], "$union$" [$\cup$], "$complement$"/ "$not$" [$\neg$], and "$bracket$" [()]. The core for determining the recipient is these set operations. To satisfy the special requirements of the graph-based namespace, we introduce a new operator "$expand$" [$\oplus$]. An expansion on a set [$\oplus(*)$] searches for all the descendants of the names in the set.

We provide several examples based on the namespace of Fig. 1. To send to a name "CA Fire" and all its descendants, we



Authorized licensed use limited to: Univ of Calif Riverside. Downloaded on October 12,2024 at 15:43:21 UTC from IEEE Xplore. Restrictions apply.

can use $\oplus$(CA Fire). This is a simple scenario in the case of the graph-based namespace. Also, we can achieve the goal of sending to multiple names with their descendants by applying the "*union*" [$\cup$] on multiple expands of names. As a second example, we show a publisher (*e.g.,* an incident commander) sending a message for all the firefighters and EMTs dealing with "Incident X". To do so, the publisher uses the formula $\oplus$(Incident X) $\cap$ ($\oplus$(Fire) $\cup$ $\oplus$(EMT)). Another example is when a publisher wants to send a message to all members of "NJ EMT" who are not dealing with "Incident X". This may happen when a commander is seeking available resources to deal with a new incident. To do so, the publisher uses the formula $\oplus$(NJ EMT) $\cap$ $\neg$ $\oplus$(Incident X) .

This allows the publisher to flexibly select recipients and is only feasible with the message-oriented solution, but not with the key-oriented solution.

### B. Efficiently Supporting Large Constant Groups

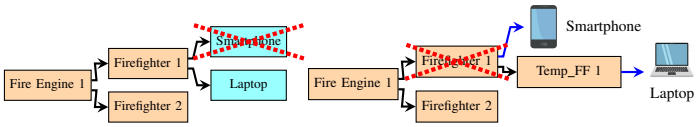
When the command chain is long and involves many names, embedding a large number of attributes/keys in the ciphertext can lead to message bloat. To address this scenario, the administrator can create a group name to represent everyone in the name hierarchy and ask each subscriber in the hierarchy to also subscribe to the new group. When a publisher wants to send a message to the old hierarchy, they can send it to the new group role instead of the old role, reducing the message size since only one attribute is needed. This mechanism does come with a penalty: the name hierarchy will be less dynamic compared to the old solution, as any change in the old name hierarchy would cause the deletion of the group name and the generation of a new group name.

## IX. Key Revocation

Key revocation allows the key issuer to deny a receiver if he is no longer trusted. It is an important feature to ensure the security of the whole system. Our solution combines timeout-based and blacklist-based mechanisms to ensure good performance. The timeout-based mechanism ensures that the receiver can only get a key for a limited period of time. To extend the period, the receiver has to renew the key from the issuer. The issuer can deny the renewal request if he believes that the receiver is no longer trusted. The timeout-based mechanism has its limitations since the key issuer can only take action when an old key expires. To overcome this limitation, our blacklist-based mechanism allows the issuer to deny a receiver at any time by placing the key in the blacklist. These two mechanisms complement each other to improve overall efficiency. Since the blacklist-based mechanism takes care of the short-term key denial, the timeout-based mechanism can give each key a relatively long lifetime and thus reduce the computation overhead for key generation. Further, since the timeout-based mechanism can invalidate each key automatically after a limited amount of time (the lifetime of the key), the blacklist-based mechanism does not have to keep the blacklist entries indefinitely. This makes the blacklist scalable over time.

### A. Timeout-based Key Revocation

To achieve Timeout-based Key Revocation, we divide time into fixed slots (*e.g.,* an hour) and use the start time of each slot as its identity. When granting a key, we encode a concatenation of the name as well as the start time of a slot into the decryption key. For example, when we grant a key for name "EMT 1" for time slot [2020-12-31 23:00:00 GMT – 2021-01-01 00:00:00 GMT), the name set $\mathbb{N}_k$ needs to contain "EMT1_t176BB078180" (the hex. representation of the epoch, valid for the 1-hour time slot). To implement this, KP-ABE and CP-ABE could embed this concatenation as an attribute in the private key and message as appropriate. For instance, assuming the underlying GUID of "EMT 1" is 0xfbc3d7, an attribute "0xfbc3d7_t176BB078180" would be embedded in the private key. For ME-PKE, each key pair should only be used in one correct time slot. When the time slot ends, NCS needs to grant a new pair of keys to the name. In response to the new key policy, the name set $\mathbb{N}_m$ would also contain this concatenation.




(a) Device Name Blacklisting (b) Temporary Name Blacklisting

Fig. 5: Key Revocation (blue arrows mean subscription)

An important property satisfied by KP-ABE, CP-ABE, and ME-PKE is that even if the private key of a time slot or a pair of keys (for ME-PKE) is compromised, an adversary cannot get the key (or decrypt messages) from a previous time slot. Our time-based private key also satisfies the Perfect Forward Secrecy property(PFS, [26], [27]). PFS (and backward secrecy) is discussed in more detail in §X.

Based on preliminary experiments (not reported here) we found that comparing the validity of time interval in the key and in the message is computationally much more complex than just comparing the time slots specified in the name of the message and the key.

The time-based key should also be able to operate even when both the senders and receivers are disconnected from the NCS for a certain period of time (meanwhile the senders and receivers can still communicate with each other via device-to-device links). If receivers (first responders) know they might need to work in an infrastructure-less environment (*e.g.,* a shelter) for a long period of time, they can prefetch (and cache) a set of keys from NCS before leaving the connected environment to cover the period they are offline. Receivers might also receive ciphertexts that were sent (and encrypted) a long time ago. The NCS needs to keep track of the authorization history of each user and have the ability to recover deleted keys. In ME-PKE, the NCS is required to store all the expired keys for potential future requests. For ABE, instead of storing expired keys, the key issuer can generate *new keys with the old timestamp*, thus eliminating the storage overhead.

### B. Blacklist-based Key Revocation

Thanks to the rich semantics provided by message-oriented encryption, a sender can easily exclude a name (via $(*) \cap \neg$(name)) when specifying the receiver name set. Blacklisting can be done on top of names instead of devices.



Authorized licensed use limited to: Univ of Calif Riverside. Downloaded on October 12,2024 at 15:43:21 UTC from IEEE Xplore. Restrictions apply.

**Device Name Blacklisting:** With blacklist-based key revocation, we mandate that each name has only 1 subscriber (or device). Blacklisting a name is equivalent to blacklisting a subscriber/device. For a first responder with multiple devices, we create unique names as children of the original name.

In Fig. 5a, we create the two *names* "Smartphone" and "Laptop" for the devices of "Firefighter 1", instead of these devices subscribing to the name "Firefighter 1". Note that these names are only required in the security layer. In the network (and information) layer, both devices still subscribe to "Firefighter 1" to maximize the benefit of multicast delivery and minimize the use of GUIDs. If the Smartphone of Firefighter 1 is stolen or lost, the key issuer can notify the sender to exclude "Smartphone", and the sender uses "$\oplus$(Fire Engine 1) $\cap$ $\neg$(Smartphone)" when encrypting the message being sent. However, this design runs counter to the principle of graph-based namespace where users focus on the roles instead of the devices (information-centricity).

**Temporary Name Blacklisting:** To address the issue with device name blacklisting, we create a temporary name for benign devices subscribing to the affected name and blacklist the original name (until all the keys the blacklisted device possesses have expired). In Fig. 5b, to blacklist "Smartphone", the NCS creates a new temporary security-layer name ("Temp_FF 1") and makes"Laptop" subscribe to the new name. From the next time slot till the end of the blacklist period, when "Laptop" requests keys for "Firefighter 1" the NCS would respond with the key for "Temp_FF 1". Thus, the receiver does not have to keep track of the name updates. At the same time, NCS needs to notify the senders of the creation of the new name, the new relationship, and the blacklist entry of the original name (and the expiration time). When sending a message, the application would use richer semantics to exclude "Firefighter 1" ("$\oplus$(Fire Fighter 1) $\cap$ $\neg$(Fire Fighter 1)"), and include "Temp_FF 1". With this, message senders do not have to keep track of namespace updates arising from blacklisting events. At the end of the blacklist period, since "Smartphone" no longer receives any key, it is safe for the NCS to revert the namespace, move "Laptop" back onto "Firefighter 1", and recycle the temp. name. However, this new solution requires the NCS to push keys to the affected receivers and the namespace updates to the senders, posing difficulties in offline/disconnected scenarios.

We get the best of both solutions by combining them: when receivers are in a connected environment, we use Temporary Name Blacklisting to minimize overhead. When receivers need to be mobilized to an infrastructure-less environment, we use Device Name Blacklisting, which slightly increases the number of names, but provides the benefit of being able to blacklist a device even without an NCS.

## X. Security Analysis

We now perform a security analysis of *SAFE* for message-oriented encryption based on KP-ABE.

**Confidentiality:** For *SAFE* only identified recipients have the ability to decrypt the message. Publishers are responsible for identifying recipients in the message. On the other hand, subscribers only hold the decryption key of the name/role they subscribe to. As a result, messages can only be decrypted by subscribers of names/roles specified as recipients in the message by the publisher.

**Integrity:** Maintaining the integrity of transmitted messages is critical to prevent attackers from modifying those messages. The integrity of a message in *SAFE* is achieved by an encrypt-then-Message Authentication Code (MAC) [33]. If any message or even the MAC is changed by a person-in-the-middle, it will be identified by the receiver who checks the MAC to validate the message.

**Security against Chosen-Plaintext Attack (CPA) & Chosen-Ciphertext Attack (CCA):** *SAFE* is CPA and CCA secure because we are building on top of KP-ABE. KP-ABE is CPA secure by design [15] and can be extended to be CCA-secure in a similar manner as in [34]. However, instead of using the transformation specified in [34], *SAFE* uses an encrypt-then-MAC procedure (signature) to ensure the authenticity of the ciphertext. With a strong unforgeable MAC, the scheme is then CCA secure. [33]

**Forward and Backward Secrecy:** In *SAFE*, each decryption key is only available for a specified time interval. The start time of the interval serves as an attribute. Any unintended key leakage of a single decryption key will enable the attacker to decrypt messages sent during only that one time period. Messages from both previous and subsequent time intervals will be secure and cannot be decrypted by attackers. Furthermore, in KP-ABE, the decryption key can only be derived from the master key, which is stored securely at the key issuer. Even with the decryption key for a specific time slot, an attacker cannot derive the decryption key for a different time slot.

**Replay Attack:** A replay attack involves an attacker capturing and repeating the same ciphertext to the receiver to exhaust the receiver's resources. To prevent this attack, a publisher could include a random nonce along with the ciphertext. Since the integrity of the message is protected by the MAC, the nonce cannot be modified during transmission. If the subscriber receives a message with a nonce that has been seen before, it will reject the message.

**Preventing Leakage of Decryption Keys:** Attackers may try to get the private key during the key distribution phase. As described in §VII, the subscriber needs to authenticate itself before requesting a decryption key from the key issuer. The key issuer encrypts the messages sent for key distribution.

TABLE III: Control Messages Needed (and Total Size using KP-ABE/CP-ABE/ME-PKE) in Different Scenarios

| **Solution** | Leaf | Subscribe | Dispatch | Delete |
|---|---|---|---|---|
| Message-Oriented | 0 | 1 (208/262/83) | 0 | 0 |
| Key-Oriented | 0 | 1 (880/566/414) | 3 (3479/2076/495) | 2 (2417/1620/96) |

## XI. Evaluation

In this section, we compare message-oriented and key-oriented encryption, and also the three encryption implementation alternatives. We use OpenABE [35] as a library for KP-ABE and CP-ABE and implementation. We found that the binary



Authorized licensed use limited to: Univ of Calif Riverside. Downloaded on October 12,2024 at 15:43:21 UTC from IEEE Xplore. Restrictions apply.

ciphertext loading function and CCA wrap function [36] in OpenABE introduces excessive overhead. For the binary ciphertext loading function, we argue that this is a library-related issue. Including it in the result may distract from understanding the performance of the core algorithm. Therefore, we excluded this part of the OpenABE library when measuring the decryption time. Additionally, *SAFE* is built on an Encrypt-then-MAC procedure (*i.e.,* adding a signature to the ciphertext), which provides CCA security [33]. Therefore, we avoid using the CCA wrap function of OpenABE. We also implemented ME-PKE utilizing the PKE API of OpenABE to provide a fair comparison across KP-ABE, CP-ABE, and ME-PKE. OpenABE PKE uses the Elliptic Curve One-Pass Diffie-Hellman algorithm. We build ME-PKE on top of that.

*SAFE* works with role-based namespaces at the application layer, and are independent of the underlying network protocol or dissemination model. We focus on metrics pertaining to the overhead of message transmission, performance, and resource usage at the application layer.

### A. Control Message Overhead

To quantify the control overheads for the message-oriented and key-oriented encryption in a dynamic environment, we compare the differences in the number of control messages required when a name change occurs. From a security perspective, control messages are only needed when new key(s) have to be delivered, such as a subscription to a new name or when the namespace changes. We present 4 scenarios based on the example namespace depicted in Fig. 1, before studying the system at scale. For simplicity, we assume that each name has only one subscriber and that initially there are no subscribers for a name. The results are shown in Table III.

**Scenario 1:** Since adding a leaf name does not affect an existing subscriber, no new keys need to be issued by the key issuer in both key-oriented and message-oriented encryption. No new control messages are required.

**Scenario 2:** Subscribing to a name, *e.g.,* after adding "Firefighter 7" to "Fire Engine 2", results in 1 control message being generated in message-oriented encryption to deliver the key of "Firefighter 7" to the new subscriber. In both KP-ABE and CP-ABE implementations, a single attribute is embedded in the new key. The size of the key is 208B (KP-ABE) or 262B (CP-ABE). For ME-PKE, the SK of "Firefighter 7" (83 bytes) is delivered to the new subscriber. In key-oriented encryption, the key in the control message needs to encode "Firefighter 7" and all its ancestors (5 names in total). Therefore, the size of the key is 880B (KP-ABE) or 566B (CP-ABE). ME-PKE delivers the SKs of all five names (size is 414 bytes).

**Scenario 3:** Dispatching "Fire Engine 1" as a child of "Search Team" (the red dashed arrow in Fig. 1) does not affect the subscribers in message-oriented encryption as the key of each subscriber only encodes the name he/she subscribes to. Thus no control messages are generated. In key-oriented encryption, each key encodes the name and all the ancestors. In this scenario, "Fire Engine 1", "Firefighter 1", and "Firefighter 2" get two new ancestors: "Incident X" and "Search Team". As a result, they each need to adjust their key(s), leading to the generation of 3 control messages in total. For KP- and CP-ABE, "Fire Engine 1" requires a new key with 6 attributes: "Fire Engine 1", "NJ Fire", "NJ", "Fire", "Search Team", and "Incident X". The key size would be 1047B (KP-ABE) or 642B (CP-ABE). Similarly, "Firefighter 1" and "Firefighter 2" each need a new key with 7 attributes since they are the children of "Fire Engine 1". Each key would be 1216B (KP-ABE) or 717B (CP-ABE). For ME-PKE, subscribers of "Fire Engine 1", "Firefighter 1", and "Firefighter 2" need the SKs of "Incident X" and "Search Team". Two new keys with a total length of 165B need to be delivered to all the 3 affected subscribers.

**Scenario 4:** relieving a team from a task. *E.g.,* removing "Fire Engine 1" from "Search Team" (reverse operation of scenario 3). Similar to scenario 3, message-oriented encryption does not need any control message for this update since the keys are not affected by the change in the relationship. In comparison, key-oriented encryption needs to "remove" the encoded "Incident X" and "Search Team" from the keys given to "Fire Engine 1", "Firefighter 1" and "Firefighter 2". With KP- and CP-ABE, the key issuer can generate new keys without the corresponding attributes, resulting in new key sizes for "Fire Engine 1" 711B (KP-ABE) or 490B (CP-ABE), and "Firefighter 1"/"Firefighter 2" 880B (KP-ABE) or 565B (CP-ABE). As for ME-PKE, the key issuer can only request the subscribes "forget" the SKs of the names. Therefore the control message size is the size of name $\times$ # of names affected (32B each). We acknowledge that this is not a proper key revocation solution for key-oriented encryption since it totally depends on the subscribers being cooperative (not using old keys in ABE and deleting SKs in ME-PKE). The real overhead for key-oriented encryption when deleting relationships in the namespace could get a lot higher, making it a very costly and inflexible solution compared to message-oriented encryption.

TABLE IV: Key-Oriented Encryption - Large Scale Simulation

| **Operations** | | 20 Dispatch | 20 Delete | 10 Dispatch + 10 Delete | 20 Subscribe |
|---|---|---|---|---|---|
| # of Control Messages | | 101 | 45 | 407 | 20 |
| Payload (KB) | KP-ABE | 202.3 | 15.2 | 340.4 | 55.9 |
| | CP-ABE | 108.0 | 14.2 | 220.7 | 33.5 |
| | ME-PKE | 42.6 | 3.5 | 30.2 | 26.6 |

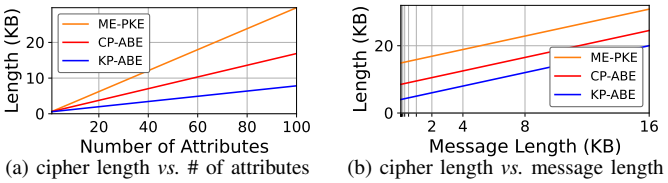
**Large Scale Simulation:** We performed a larger-scale simulation to study the control message overheads (key-oriented encryption) and the size of the ciphertext (both cases).

*Simulation setup:* We simulate two hierarchical namespaces: an administrative hierarchy and an incident hierarchy. The administrative tree has a randomly generated number of levels and a random number of branches, both ranging from 4 to 6, resulting in a total of 1337 nodes. The incident tree is created based on the level and fanout ranging from 3 to 5, comprising a total of 155 nodes. We assume each name has only one subscriber. During the simulation, we observed the control messages generated and their payloads for randomly generated operations. Results are in Table IV.

*Simulation results:* First, we randomly selected 20 nodes from the administrative tree and dispatch them as children of randomly chosen nodes in the incident tree. This resulted in the generation of 101 control messages, indicating that not only



Authorized licensed use limited to: Univ of Calif Riverside. Downloaded on October 12,2024 at 15:43:21 UTC from IEEE Xplore. Restrictions apply.



(a) cipher length *vs.* # of attributes (b) cipher length *vs.* message length

Fig. 6: Message-Oriented Encryption - Ciphertext Length(KB).

the selected nodes but also their descendants were affected. Table IV shows that KP-ABE sends about 2 × the payload of CP-ABE to deliver new keys, while MK-PKE is much lower. In the second scenario, we randomly deleted 20 relationships from the administrative tree. This led to the generation of 45 control messages. The relatively small amount of overhead is due to the selection of relatively low-level relationships that were deleted. The third scenario combined the first two scenarios. We first dispatch 10 nodes from the administrative tree as children of nodes in the incident tree and then deleted 10 relationships in the administrative tree. This resulted in the generation of 407 control messages. Again, KP-ABE sends more bytes than the others, with ME-PKE introducing much lower overhead. The relatively large amount of overhead is due to the selection of relatively higher-level relationships. In the last scenario, we randomly selected 20 nodes from the administrative tree and let each one have a new subscriber. This resulted in a total of 20 control messages, as we only needed to deliver the new key(s) to the new subscribers. Again, KP-ABE is more expensive. Overall, from all the scenarios, KP-ABE had the largest payloads, followed by CP-ABE, and ME-PKE had the lowest payloads.

### *B. Ciphertext Length*

To evaluate the overhead for sending messages in both encryptions. We first generally compare different implementations and then evaluate them with a large-scale simulation.

In Fig. 6 we show the length of ciphertext (with signature) for a 16-byte attribute length which is large enough to identify each group addressed in the communication, of KP-ABE, CP-ABE, and ME-PKE with message-oriented encryption. In Fig. 6a, we show the ciphertext length for a varying number of attributes, for a 160-byte message, which is a typical SMS message [37]. The number of attributes are selected based on the expected number of recipients for typical messages during disaster communication. Fig. 6b, shows the ciphertext length for varying message sizes, with a fixed number of 50 attributes. We believe the range of the message size used here covers the range of sizes for typical text-, image-, and voice-based messages in disaster communication. Both results show that both KP-ABE and CP-ABE, as well as ME-PKE, grow linearly, with KP-ABE having the lowest rate of increase. KP-ABE has the shortest ciphertext, followed by CP-ABE. In KP-ABE, the set of attributes is included in the ciphertext. For CP-ABE, the access structure is included in the ciphertext. The access structure is longer than the set of attributes because it not only contains attributes but also their relationships. In ME-PKE, we use a 32-byte (256-bits) randomly generated key for each receiver, which is larger than the 16-byte attribute, resulting in the largest ciphertext size.

TABLE V: Total Ciphertext Length (KB) for Sending Message

| # of names sending to | | 10 | 20 | 30 |
|---|---|---|---|---|
| Message-Oriented | KP-ABE | 7.3 | 18.6 | 22.5 |
| | CP-ABE | 9.9 | 28.5 | 30.7 |
| | ME-PKE | 12.8 | 41.0 | 40.1 |
| Key-Oriented | KP-ABE | 5.7 | 11.5 | 17.3 |
| | CP-ABE | 6.3 | 12.7 | 19.1 |
| | ME-PKE | 6.4 | 12.9 | 19.3 |

For large-scale simulation for both message and key-oriented encryption, we used the previously generated administrative tree as the target. We randomly chose 10, 20, and 30 names to send a message to and calculate the total ciphertext length (results in Fig. 6). There is a higher probability of selecting lower-level nodes (more of them) with fewer descendants compared to higher-level nodes. While this may slightly diminish the advantages of key-oriented encryption, it is reasonably typical of real-life scenarios where communication occurs more frequently between sub-departments or individual first responders rather than the entire department.

Table V presents the simulation results. Key-oriented encryption generally results in shorter ciphertext lengths compared to message-oriented encryption. The overhead in KP-ABE for message-oriented encryption is less than 1.5 times that of key-oriented encryption in each scenario. CP-ABE and ME-PKE have larger ciphertexts than KP-ABE.

### *C. Comparison among Options*

Based on the analysis of control messages in different scenarios, it is evident that message-oriented encryption offers advantages in terms of control message requirements and key size compared to key-oriented encryption. This is particularly beneficial in dynamic environments where frequent name changes occur, such as the addition or deletion of incidents or teams. Furthermore, the size of keys in key-oriented encryption, especially in implementations like KP-ABE and CP-ABE, can become significantly larger with deeper name hierarchies. Also, key-oriented encryption needs to revoke the old key each time the namespace is updated, which will be complex in a dynamic environment. When comparing the overhead of sending messages, it is observed that key-oriented encryption generates shorter ciphertext compared to message-oriented encryption in all implementations. However, in the case of KP-ABE, the reduction in ciphertext length is no more than a factor of 1.5, which diminishes the advantage of the key-oriented solution. In comparison to the payload for name updating, we aim to demonstrate that this price is worth paying. Based on these findings, we recommend using message-oriented encryption as the preferred approach.

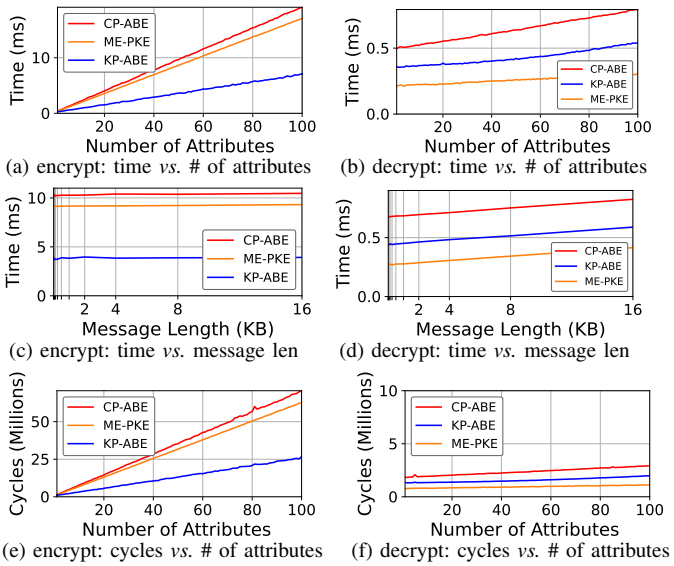
### *D. Comparison among Encryption Schemes*

We now compare KP-ABE, CP-ABE, and ME-PKE implementations based on message-oriented encryption.

**Key Generation:** Based on our experiment, in message-oriented encryption, the time taken for a single key generation is 0.28 milliseconds (ms) for CP-ABE, 0.73 ms for KP-ABE, and 0.01 ms for ME-PKE. It should be noted that in message-oriented encryption, each ABE key only has one attribute, and each name has only one PK/SK pair. Therefore, we present



Authorized licensed use limited to: Univ of Calif Riverside. Downloaded on October 12,2024 at 15:43:21 UTC from IEEE Xplore. Restrictions apply.



(a) encrypt: time *vs.* # of attributes (b) decrypt: time *vs.* # of attributes
(c) encrypt: time *vs.* message len (d) decrypt: time *vs.* message len
(e) encrypt: cycles *vs.* # of attributes (f) decrypt: cycles *vs.* # of attributes

Fig. 7: Encryption and Decryption Performance.

the results with one attribute. We set the attribute length to 16 bytes in both KP-ABE and CP-ABE. This experiment and the following experiment in this section were conducted on a system with 12 CPU cores and 32 GB memory.

The results indicate that KP-ABE takes the longest time for key generation, followed by CP-ABE, while ME-PKE is the fastest. In KP-ABE, the access structure is embedded in the key, while in CP-ABE, the attribute list is embedded in the key. Therefore, CP-ABE is expected to be faster than KP-ABE for key generation. Generating PK/SK key pairs is the fastest process, as it does not embed any attributes.

**Encryption Performance:** We now compare the encryption performance of KP-ABE, CP-ABE, and ME-PKE, measuring the running time for each encryption scheme with a 16-byte attribute length, and message length of 160 bytes. Figure 7a shows that the encryption time for all three encryption schemes grows almost linearly as the number of attributes increases.

Figure7c shows the running time as the message length increases, with the number of attributes fixed at 50. The three schemes grow slowly with the message length. Only when the message size grows to the order of MBytes, the encryption time starts to increase substantially (*e.g.,* images and video).

KP-ABE has the best encryption performance among all three schemes. It has a set of attributes embedded in the ciphertext, while CP-ABE needs to create an access structure which has more impact on performance. For ME-PKE, Open-ABE PKE uses the Elliptic Curve One-Pass Diffie-Hellman algorithm. Generating a Diffie-Hellman common shared key to encrypt a message for each receiver based on their PK (and thus MK-PKE) is expensive.

**Decryption Performance:** We evaluated the running time of decryption for KP-ABE, CP-ABE, and ME-PKE in the message-oriented solution.

Figure 7b shows the decryption time varying with the number of attributes for message-oriented encryption. We still use 16 bytes as the length of an attribute and a 160-byte message. Each subscriber of a name holds a key with one attribute (KP-ABE, CP-ABE) or one SK (ME-PKE). The number of attributes here indicates how many names were embedded in the encrypted message. Again, the decryption time is relatively flat. The variability we observe in the plots are typical measurement-related variations, based on the task being repeated 100 times.

Thus, see that ME-PKE has the lowest result decryption time, with KP-ABE being the second best. The reason is that symmetric key decryption in ME-PKE is faster. CP-ABE is slower because with message-oriented encryption, CP-ABE needs to recover a long access structure in the message. On the other hand, KP-ABE only has an access structure with a single attribute in the key.

**Resource usage:** Figures 7e and 7f shows the CPU consumption (millions of cycles) for each encryption and decryption operation, as the number of attributes is varied. It tracks the encryption and decryption time results. The attribute length is 16 bytes and the message size is 160 bytes. During encryption, CP-ABE has the highest CPU cycle consumption, followed by ME-PKE, with KP-PKE using the fewest number of CPU cycles. For decryption, CP-ABE still has the highest CPU cycle consumption, followed by KP-ABE, while ME-PKE has the lowest CPU consumption. Regarding memory footprint, CP-ABE, KP-PKE, and ME-PKE are all close to each other and increase linearly. Specifically, when embedding 50 attributes/names, all three schemas have similar behavior, consuming approximately 488 MB during the encryption process and around 494 MB for decryption.

## XII. Conclusion

In this paper, we designed *SAFE*, an encryption mechanism for a name-based disaster pub/sub communication system to achieve a flexible, scalable multiparty communication and efficient data distribution while still maintaining confidentiality and integrity. We abstract the encryption layer, supporting either message-oriented encryption or key oriented-encryption. The abstract encryption layer could be implemented by multiple basic encryption schemes, including KP-ABE, CP-ABE, and ME-PKE. Based on our evaluation of our implementation as well as simulations, KP-ABE has the shortest ciphertext length and lowest encryption time, although it is slower for key generation and decryption. However, since message-oriented encryption does not require new keys to be generated on namespace updates, the key generation process is much less frequent, mitigating this concern for KP-ABE. We avoid ME-PKE because of its poor encryption performance. Another concern with ME-PKE is that publishers need to hold all the public keys of the names they want to send to. Even for namespace updates, KP-ABE and CP-ABE require a single public key to handle all encryption. *SAFE* uses message-oriented encryption with KP-ABE for first-responder communication.

## XIII. Acknowledgement

This work was supported by the US Department of Commerce, NIST (award 70NANB17H188) and US NSF grant CNS-1818971. We thank our shepherd, Prof. Spyros Mastorakis, for his support and the reviewers for their valuable comments.



Authorized licensed use limited to: Univ of Calif Riverside. Downloaded on October 12,2024 at 15:43:21 UTC from IEEE Xplore. Restrictions apply.

## REFERENCES


[1] L. Zhang, A. Afanasyev, J. Burke, V. Jacobson, K. Claffy, P. Crowley, C. Papadopoulos, L. Wang, and B. Zhang, "Named data networking," *ACM SIGCOMM Computer Communication Review*, vol. 44, no. 3, pp. 66–73, 2014.

[2] D. Raychaudhuri, K. Nagaraja, and A. Venkataramani, "Mobilityfirst: a robust and trustworthy mobility-centric architecture for the future internet," *ACM SIGMOBILE Mobile Computing and Communications Review*, vol. 16, no. 3, pp. 2–13, 2012.

[3] A. Hannan, S. Arshad, M. A. Azam, J. Loo, S. H. Ahmed, M. F. Majeed, and S. C. Shah, "Disaster management system aided by named data network of things: Architecture, design, and analysis," *Sensors*, vol. 18, no. 8, p. 2431, 2018.

[4] M.-N. Tran and Y. Kim, "Named data networking based disaster response support system over edge computing infrastructure," *Electronics*, vol. 10, no. 3, p. 335, 2021.

[5] N. Fotiou, P. Nikander, D. Trossen, and G. C. Polyzos, "Developing information networking further: From psirp to pursuit," in *Broadband Communications, Networks, and Systems: 7th International ICST Conference, BROADNETS 2010, Athens, Greece, October 25–27, 2010, Revised Selected Papers 7*. Springer, 2012, pp. 1–13.

[6] J. Chen, M. Arumaithurai, L. Jiao, X. Fu, and K. Ramakrishnan, "Copss: An efficient content oriented publish/subscribe system," in *2011 ACM/IEEE Seventh Symposium on Architectures for Networking and Communications Systems*. IEEE, 2011, pp. 99–110.

[7] M. Jahanian, J. Chen, and K. K. Ramakrishnan, "Graph-based namespaces and load sharing for efficient information dissemination in disasters," in *2019 IEEE 27th International Conference on Network Protocols (ICNP)*. IEEE, 2019.

[8] J. Chen, M. Arumaithurai, X. Fu, and K. Ramakrishnan, "Cns: content-oriented notification service for managing disasters," in *Proceedings of the 3rd ACM Conference on Information-Centric Networking*, 2016.

[9] Y. Yu, A. Afanasyev, D. Clark, K. Claffy, V. Jacobson, and L. Zhang, "Schematizing trust in named data networking," in *proceedings of the 2nd ACM Conference on Information-Centric Networking*, 2015, pp. 177–186.

[10] R. L. Rivest, A. Shamir, and L. Adleman, "A method for obtaining digital signatures and public-key cryptosystems," *Communications of the ACM*, vol. 21, no. 2, pp. 120–126, 1978.

[11] D. Brown, "Standards for efficient cryptography, sec 1: elliptic curve cryptography," *Released Standard Version*, vol. 1, 2009.

[12] A. Shamir, "Identity-based cryptosystems and signature schemes," in *Workshop on the theory and application of cryptographic techniques*. Springer, 1984, pp. 47–53.

[13] A. Sahai and B. Waters, "Fuzzy identity-based encryption," in *Advances in Cryptology–EUROCRYPT 2005: 24th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Aarhus, Denmark, May 22-26, 2005. Proceedings 24*. Springer, 2005, pp. 457–473.

[14] M. Ion, J. Zhang, and E. M. Schooler, "Toward content-centric privacy in icn: Attribute-based encryption and routing," in *In Proceedings of the 3rd ACM SIGCOMM workshop on Information-centric networking*, 2013, pp. 39–40.

[15] V. Goyal, O. Pandey, A. Sahai, and B. Waters, "Attribute-based encryption for fine-grained access control of encrypted data," in *Proceedings of the 13th ACM conference on Computer and communications security*, 2006, pp. 89–98.

[16] J. Bethencourt, A. Sahai, and B. Waters, "Ciphertext-policy attribute-based encryption," in *2007 IEEE symposium on security and privacy (SP'07)*. IEEE, 2007, pp. 321–334.

[17] V. Jacobson, D. K. Smetters, J. D. Thornton, M. F. Plass, N. H. Briggs, , and R. L. Braynard, "Networking named content," in *In Proceedings of the 5th international conference on Emerging networking experiments and technologies*, 2009, pp. 1–12.

[18] J. Seedorf, A. Tagami, M. Arumaithurai, Y. Koizumi, N. B. Melazzi, D. Kutscher, K. Sugiyama, and et al., "The benefit of information centric networking for enabling communications in disaster scenarios," in *In 2015 IEEE Globecom Workshops (GC Wkshps)*. IEEE, 2015, pp. 1–7.

[19] S. Mukherjee, F. Bronzino, S. Srinivasan, J. Chen, and D. Raychaudhuri, "Achieving scalable push multicast services using global name resolution," in *In 2016 IEEE Global Communications Conference (GLOBECOM)*. IEEE, 2016, pp. 1–6.

[20] C. Gündoğan, P. Kietzmann, T. C. Schmidt, and M. Wählisch., "Hopp: Robust and resilient publish-subscribe for an information-centric internet of things," in *In 2018 IEEE 43rd Conference on Local Computer Networks (LCN)*. IEEE, 2018, pp. 331–334.

[21] A. Shariat, A. Tizghadam, and A. Leon-Garcia, "An icn-based publish-subscribe platform to deliver uav service in smart cities," in *2016 IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS)*. IEEE, 2016, pp. 698–703.

[22] B. Segall, D. Arnold, J. Boot, M. Henderson, and T. Phelps, "Content based routing with elvin4," in *In Proceedings of AUUG2K*, no. 39, 2000.

[23] Y. Diao, S. Rizvi, and M. J. Franklin, "Towards an internet-scale xml dissemination service," in *In Proceedings of the Thirtieth international conference on Very large data bases-Volume 30*. IEEE, 2004, pp. 612–623.

[24] R. Canetti, J. Garay, G. Itkis, D. Micciancio, M. Naor, and B. Pinkas, "Multicast security: A taxonomy and some efficient constructions," in *In IEEE INFOCOM'99. Conference on Computer Communications. Proceedings. Eighteenth Annual Joint Conference of the IEEE Computer and Communications Societies. The Future is Now (Cat. No. 99CH36320)*, 1999, pp. 708–716.

[25] AT&T, "Firstnet website," https://www.firstnet.com/, [accessed on Ayg. 19, 2023].

[26] W. Diffie, P. C. Van Oorschot, and M. J. Wiener, "Authentication and authenticated key exchanges," *Designs, Codes and cryptography*, vol. 2, no. 2, pp. 107–125, 1992.

[27] C. G. Günther, "An identity-based key-exchange protocol," in *Advances in Cryptology—EUROCRYPT'89: Workshop on the Theory and Application of Cryptographic Techniques Houthalen, Belgium, April 10–13, 1989 Proceedings 8*. Springer, 1990, pp. 29–37.

[28] K. Yongdae, A. Perrig, and G. Tsudik, "Simple and fault-tolerant key agreement for dynamic collaborative groups," in *Proceedings of the 7th ACM Conference on Computer and Communications Security*, 2000.

[29] K. Teemu, M. Chawla, B.-G. Chun, A. Ermolinskiy, K. H. Kim, S. Shenker, and I. Stoica, "A data-oriented (and beyond) network architecture," in *In Proceedings of the 2007 conference on Applications, technologies, architectures, and protocols for computer communications*, 2007, pp. 181–192.

[30] K. K. Ramakrishnan, M. Yuksel, H. Seferoglu, J. Chen, and R. A. Blalock, "Resilient communication for dynamic first responder teams in disaster management," *IEEE Communications Magazine*, vol. 60, no. 9, pp. 93–99, 2022.

[31] OpenSSL Technical Committee, "The openssl project," https://www.openssl.org/, [accessed on May. 9, 2023].

[32] J. Katz and Y. Lindell, *Introduction to modern cryptography*. CRC press, 2020, ch. 10.3, pp. 330–338.

[33] M. Bellare and C. Namprempre, "Authenticated encryption: Relations among notions and analysis of the generic composition paradigm," in *Advances in Cryptology—ASIACRYPT 2000: 6th International Conference on the Theory and Application of Cryptology and Information Security Kyoto, Japan, December 3–7, 2000 Proceedings 6*. Springer, 2000, pp. 531–545.

[34] S. Amit, "Non-malleable non-interactive zero knowledge and adaptive chosen-ciphertext security," in *In 40th Annual Symposium on Foundations of Computer Science (Cat. No. 99CB37039)*. IEEE, 1999, pp. 543–553.

[35] Zeutro, LLC, "The openabe library," https://github.com/zeutro/openabe/, [accessed on May. 9, 2023].

[36] ——, "The openabe design document," https://github.com/zeutro/openabe/blob/master/docs/libopenabe-v1.0.0-design-doc.pdf, [accessed on May. 9, 2023].

[37] E. Wilde, "Uri scheme for global system for mobile communications (gsm) short message service (sms)," Internet Engineering Task Force (IETF), RFC Editor, RFC 5724, 1 2010. [Online]. Available: https://www.ietf.org/rfc/rfc5724.txt




Authorized licensed use limited to: Univ of Calif Riverside. Downloaded on October 12,2024 at 15:43:21 UTC from IEEE Xplore. Restrictions apply.