TKBE : TWO KEY BROADCAST ENCRYPTION FOR THE IOT

A Dissertation Submitted

in Partial Fulfilment of the Requirements

for the Degree of

Master of Technology

in

Cryptology and Security

by

RACHIT PARIKH

Under the joint supervision of

Prof. Bart Preneel

Prof. Bimal Kumar Roy

ISI KOLKATA

COSIC, KU LEUVEN

July, 2023

ACKNOWLEDGEMENT

I am immensely grateful to **Prof. Bart Preneel** and **Prof. Bimal Kumar Roy** for their joint supervision and unwavering support throughout my internship at COSIC KU Leuven. I am thankful to Prof. Bart Preneel for providing me with the opportunity to work at COSIC KU Leuven, and Prof. Bimal Kumar Roy for recommending me.

I would like to extend my heartfelt thanks to **Dr. Dave Singelee** and **Sayon Duttagupta** for their constant guidance and assistance throughout the thesis. Their expertise, patience, and willingness to address my queries and challenges have played a crucial role in overcoming obstacles and refining my work. I would also extend my thanks to COSIC, KU Leuven for providing me with the opportunity.

Furthermore, I want to express my gratitude to **Pratyay Mukherjee**, a visiting scientist at ISI Kolkata, for his timely support and assistance when I encountered a challenging problem in my research. I would like to extend my heartfelt appreciation to **Prof. Subhamoy Maitra** for his significant contributions to my research experience and his invaluable guidance in understanding the methodology of research. Finally, I am thankful to my esteemed teachers and mentors at **Indian Statistical Institute Kolkata** who have played a pivotal role in shaping my academic journey and retaining my interest in Cryptology throughout my Master's degree.

Leuven, Belgium **Rachit Parikh**

ABSTRACT

The growing usage of the Internet of Things (IoT) has made it necessary to ensure the security of these interconnected devices. Key management becomes particularly challenging when devices are not always online due to resource constraints or business decisions. Moreover, the IoT infrastructure typically relies on the publish-subscribe model for communication, which raises additional security considerations since the message broker becomes a central point of attack. Existing solutions with end-to-end encryption from publisher to subscriber are either computationally expensive for resource constrained devices or compromise on the decoupling in publish/subscribe systems. This thesis tackles the problem of efficient key management in IoT systems by employing techniques from broadcast encryption and proposes a lightweight framework - TKBE (Two Key Broadcast Encryption) that reduces trust in the broker and enhances security in IoT communications while providing efficient immediate revocation with decoupling and offline key updates.

Contents

[1]Exactly the same algorithms are invoked on addition of a user

NOTATION

Symbol	Description
$\texttt{Path}(x)$	Returns set of nodes in path from x to root excluding x
$s(x)$	Sibling of node x in a binary tree
$p(x)$	Parent of node x in a binary tree
$\texttt{Enc}(M, K)$	Encryption of message M under key K
$E(\mathbb{F}_q)$	Elliptic Curve
G	Generator point
q	Schnorr signature modulus
$KDF(.)$	Key Derivation Function
$H(.)$	Hash function
$\in^R$	Randomly generated
gk_i	Group key after i-revocations/additions
tk_i	Token key after i-revocations/additions
$time$	Current time
$time_i$	Timestamp during i-th revocation/addition

DEDICATED TO

The Resilience of KDF Chains

1 Introduction

The Internet of Things (IoT) describes a network of devices or *things* with processing abilities that can connect with other devices for data exchange over a network. The term *Internet of Things* first appeared in a speech by Peter Lewis addressed to the Congressional Black Caucus Foundation 15th Annual Legislative Weekend in Washington, D.C., in 1985 [Lak+20]. These *things* range from large-scale industrial equipment to the thermostats in smart homes. By 2025, it is predicted that the number of connected devices will exceed 75 billion [Fou20].

IoT technology has been widely adopted across various domains, including home automation, agriculture, healthcare, smart vehicles, and more. Particularly in industries, it is extensively utilized as the Industrial Internet of Things (IIoT) to monitor and control industrial systems, optimize manufacturing processes, and enable smart grid energy optimization, among other applications [Xu+18]. Additionally, the military sector has also embraced IoT technology, as evidenced by recent initiatives such as the Internet of Battlefield Things (IoBT) project initiated by the U.S. Army Research Laboratory (ARL). The IoBT project focuses on leveraging IoT applications to enhance the capabilities of soldiers in the field [ZME21].

Given the pervasive nature of IoT technology, addressing security and privacy concerns has become crucial. However, the diverse range of IoT devices, each with varying computing capabilities and communication modes, presents challenges in developing standardized guidelines and regulatory frameworks. There are several architectures for communication in the IoT. For example, XMPP, CoAP, MQTT are some protocols for application layer security [Kar+15]. Among these, MQTT is widely used, for it being an extremely lightweight publish/subscribe messaging transport that is ideal for connecting remote devices with a small code footprint and minimal network bandwidth [MQT].

The publish/subscribe messaging system handles most of the information that is being transmitted around the world via Internet [LTP22]. In this system, three entities coexist: the message broker, the publisher, and the subscriber. The subscriber registers or subscribes its interests with the broker, indicating the topics or areas of information it is interested in. Similarly, the publisher registers or publishes its interests with the broker, specifying the topics or areas it will provide information on. The broker (might be collection of brokers as shown in Figure 1.1) acts as an intermediary, managing the flow of information by forwarding data related to a particular topic from the publisher to the subscriber interested in that topic. This way, the publisher and subscriber are decoupled. This property of *decoupling* is crucial for scalability of a system. The subscriber or the publisher are not aware of each other's existence in this system [MQT].

However, in publish/subscribe protocols like [Mal+19] the inherent assumption of trust in the broker lets an eavesdropping adversary to get hold of sensitive data that is meant only to be seen by the subscribers. This calls for a requirement of an end-

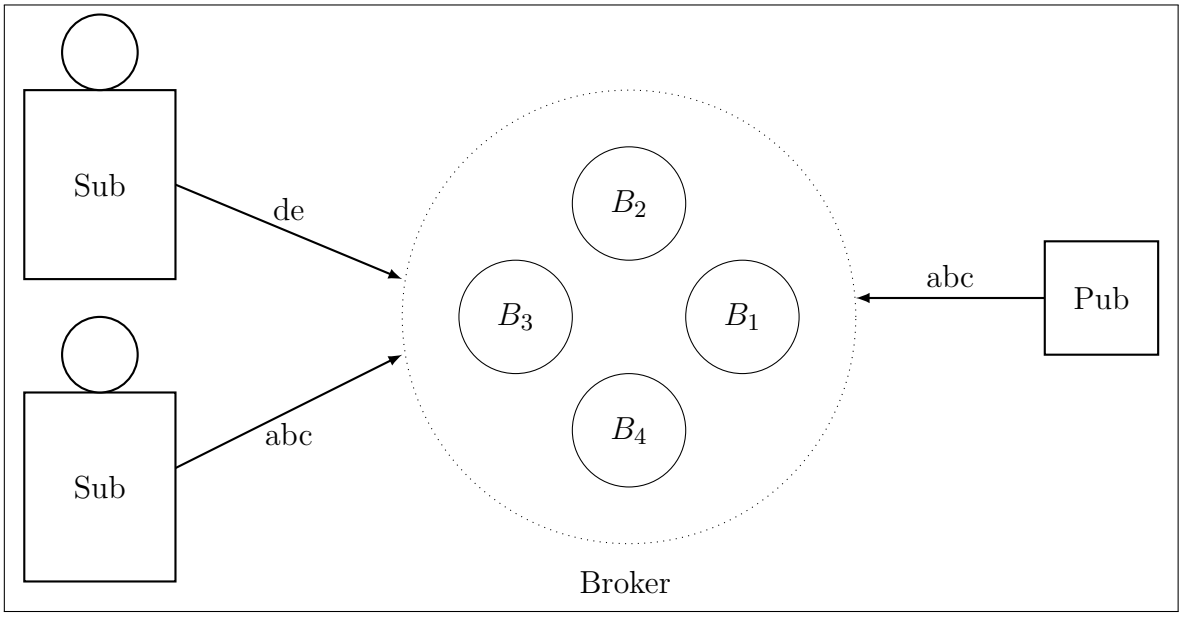

Figure 1.1: Publish-Subscribe messaging system with Publisher publishing on topic 'abe' and two subscribers subscribed to topics 'de' and 'abc'

to-end encryption protocol between subscribers and publishers. When implementing end-to-end encryption protocols in a publish/subscribe model for the IoT, there are challenges that arise due to the need for the sender to know the public key of the receiver. This requirement can introduce inefficiencies and hinder the scalability of the system, particularly when resource-constrained IoT devices are involved. Additionally, integrating techniques like Attribute Based Encryption (ABE) or Proxy Re-Encryption (PRE) to address these challenges may come with compromises in terms of decoupling and feasibility for resource constrained devices as they reduce the battery life of these devices [Kum+19; Pol+17].

This thesis focuses on enhancing the security of IoT environments, specifically for the publish/subscribe systems where the broker is honest but curious. In this sce-

nario, an adversary can eavesdrop the communication between broker and clients. The protocol incorporates a token manager and utilizes a two-step encryption process to ensure data confidentiality. Firstly, the message is encrypted using a shared token that is updated asynchronously between the publisher and the token manager. Then, the encrypted message is published to the broker, which further re-encrypts the message to provide an additional layer of security. The broker then broadcasts the doubly-encrypted message to the subscribers. By employing this two-layer encryption approach, the protocol enhances data confidentiality within the IoT system. It operates under the assumption that the broker and token manager do not collude, ensuring that even if one entity is compromised, the data remains protected due to the additional layer of encryption provided by the other entity. The protocol developed in this work aims to achieve the following objectives:

- **Data confidentiality**. The protocol safeguards the exchanged data by encrypting it using two keys, preventing unauthorized access or eavesdropping by the broker to read contents of the data.
- **Lightweight protocol with minimal computational and communication overhead**. The proposed protocol uses only one group key per topic from the broker side rather than encrypting for each subscriber with a different key, reducing the overhead. It also uses LKH (Logical Key Hierarchy) for efficient key management [WHA99].
- **Implement an efficient revocation mechanism while maintaining decoupling**. The protocol also incorporates an efficient revocation mechanism that allows for prompt and secure revocation of nodes without compromising the decoupling between message receivers and senders. Revocation also

happens from the token manager side whenever a new subscriber is added or revoked, the tokens are asynchronously updated by both the publisher and the token manager.

- **Reduce reliance on trust in the broker**. The confidentiality of information published is preserved even if an adversary observes the information passed through the broker, has access to the group keys unless it colludes with the token manager.

First, I discuss work related to publish/subscribe in the IoT and some existing protocols in Literature Review, then preliminaries that will be used in TKBE will be covered in Background. The protocol will be discussed in Protocol, followed by details related to security and implementation in Conclusion. Finally, I discuss some future work related to authorization framework and publish/subscribe protocols in Future Work.

2 Literature Review

The popular publish/subscribe frameworks such as MQTT use TLS/SSL for security which are not very efficient. The work presented in [Kat+17] utilizes AES with MAC for encryption, but lacks end-to-end encryption at the application layer. Additionally, the protocol is computationally expensive for resource-constrained devices. The work in [Ahm+17] discusses that lightweight payload encryption scheme over CoAP (Constrained Application Protocol) enforces Datagram Transport Layer Security. Additionally, surveys like [Mal+16; Xu+18; Kha+22] provide a comparison of protocols, challenges and categorization of approaches used in IoT security.

JEDI (Joining Encryption and Delegation for IoT) [Kum+19] is a many-to-many end to end encryption protocol that does not rely on central trust. It is compatible with decentralized authorization frameworks like [And+19]. It uses WKD-IBE [AKN08] which is a hierarchical identity based encryption scheme that uses wildcard key derivations. JEDI provides decentralized delegation, decoupling and end to end encryption for resource constrained devices. Their notable optimization in WKD-IBE makes it appropriate to use for resource constrained devices. However, the benchmarks have noted reduction in battery life even after periodic use of IBE. While it provides decoupling between senders and receivers, during revocation, the

senders will need to know the identity of the revoked users which in turn affects the decoupled nature of the publish/subscribe communication. Other Attribute Based Encryption schemes like [Goy+06; Wan+16; BSW07] are not as efficient and slower than JEDI.

Malina et al. [Mal+19] proposed a secure publish subscribe framework with three layers of security providing different security guarantees. With security level 3 giving the most, data integrity, non-repudiation and data privacy. However, the Broker in this protocol can read the data published to it. Also, the broker has to create a session with each subscriber before sending a message. However, the protocol also provides an option of multicast communication, reducing the number of encryption from the broker side. The protocol uses a trusted revocation authority for revocation which again can be a problem if the honest but curious broker colludes with a revoked user. The protocol proposed in this thesis will be using primitives and notation from this work, like `AES-GCM-SIV`, `ECIES`.

TooLate [Bra+21] protocol is specifically meant for offline devices and key management in offline devices. There are three entities in this scenario where the key management system and the offline data storage agree on a random salt before the initialization of protocol and create a hash chain based on time-slots and salt, being in synchronization without ever interacting again. This protocol can be used for extremely low powered devices or *zero-state* devices that have bandwidth issues. This thesis uses the idea of time synchronization by sharing a random value and then refreshing key using it. The token manager and publisher in TKBE will be also sharing a random salt before the protocol initiates and creating a hash chain to encrypt the tokens. Proof for protection against expired keys, key forging resistance

can be found in [Bra+21].

From the above protocols and literature review it is clear that among the publish/subscribe protocols meant for MQTT, there is no protocol that is lightweight with reduced trust in the broker, an efficient revocation mechanism and still maintains decoupling. The protocol proposed in this thesis achieves these goals and other security goals that are mentioned in Conclusion.

3 Background

This chapter provides preliminaries related to the protocol. It will skip certain basic cryptography primitives like Hash functions [BS20], `AES-GCM-SIV` (Nonce Misuse-Resistant Authenticated Encryption) [GLL19], Schnorr signature scheme [Sch90] and Elliptic Curve Integrated Encryption Scheme (`ECIES`) [Sho01].

3.1 Publish/Subscribe System

As discussed earlier, the publish/subscribe consists of three entities: publisher, a broker and subscriber. In a publish/subscribe system, the publisher and subscriber are unaware about each other's existence. This provides decoupling in terms of time, space and synchronization. The architecture provides several types of message filtering options, being topic-based, content-based and type-based. In this thesis, I will be working on topic-based filtering. One such example is shown in Figure 3.1.

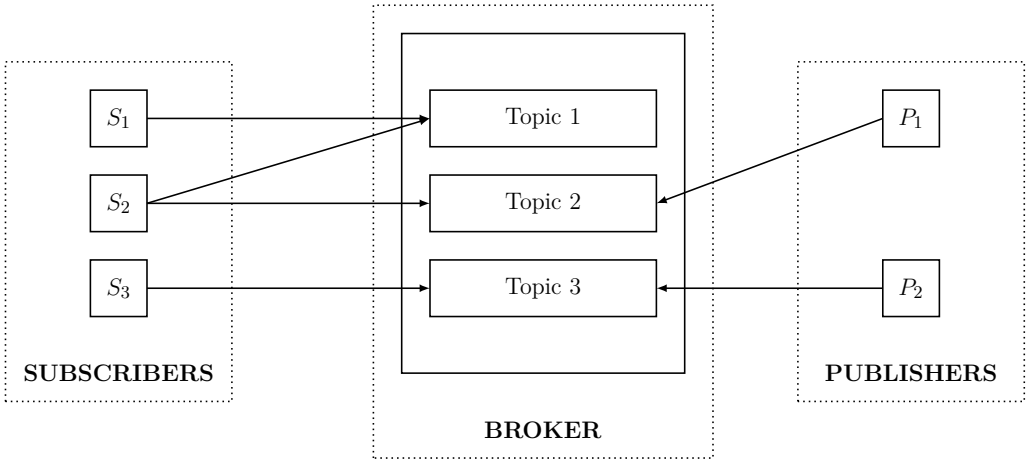

Figure 3.1: Publish/Subscribe system

Here, subscriber S_1 is subscribed to the topic *Topic 1*, subscriber S_2 is subscribed to two topics *Topic 1* and *Topic 2*, and subscriber S_3 is subscribed to *Topic 3*. The job of the broker is to forward any message labelled with *Topic 1* to the subscribers S_1 and S_2. The topic based filtering is internally done by the broker. The advantages of using such system are increased scalability, loose coupling, many to many communication, and increased fault tolerance. A more detailed survey about Publish/Subscribe systems for IoT can be found in [LTP22].

The MQTT protocol is widely adopted for pub/sub messaging, offering a scalable and reliable means to connect devices over the internet. It has found significant applications in the IoT, with notable usage by platforms like Facebook Messenger [BS18]. MQTT boasts a lightweight footprint, minimizing energy consumption and computation requirements. This makes it well-suited for resource-constrained IoT devices, as it helps preserve battery life and reduces bandwidth usage.

However, MQTT does face security challenges in terms of robustness and efficiency. While it incorporates SSL/TLS for transport security, the computational overhead and memory usage associated with TLS can be significant for resource-constrained IoT devices [Mal+19]. Additionally, MQTT relies on X.509 client certificates for authentication, which can incur additional costs. Furthermore, the broker controls the authorization of topics in MQTT, determining which subscribers are allowed to subscribe to specific topics.

3.2 LKH Scheme

To enhance efficiency in a pub/sub system, it is advantageous for a broker to employ a group key for subscribers, ensuring that only authorized members can decrypt messages pertaining to a specific topic. By organizing subscribers into groups based on their topic subscriptions, the broker can encrypt messages using a single group key and broadcast them exclusively to the authorized recipients.

However, several challenges arise when implementing this approach, particularly regarding revoked users who still possess the group key. To address this issue, a potential solution is to update the group key whenever a user is revoked or added. This practice prevents added users from accessing previous messages and ensures that revoked users cannot decrypt future messages. Nonetheless, continuously sending individual key updates to all users can lead to inefficiencies in key management.

To encounter this problem, Wallner et al. [WHA99] proposed a key management scheme LKH (Logical Key Hierarchy) for multicast communications. Along with initializing a common group key, this method also provides an efficient way to rekey the multicast group. For a revocation, rather than individually updating the group key, the broker can simply broadcast $\mathcal{O}(\log N)$ messages (where N is the number of subscribers) for key updates, thus minimizing the number of transmissions for group re-keying and the storage for multicast groups.

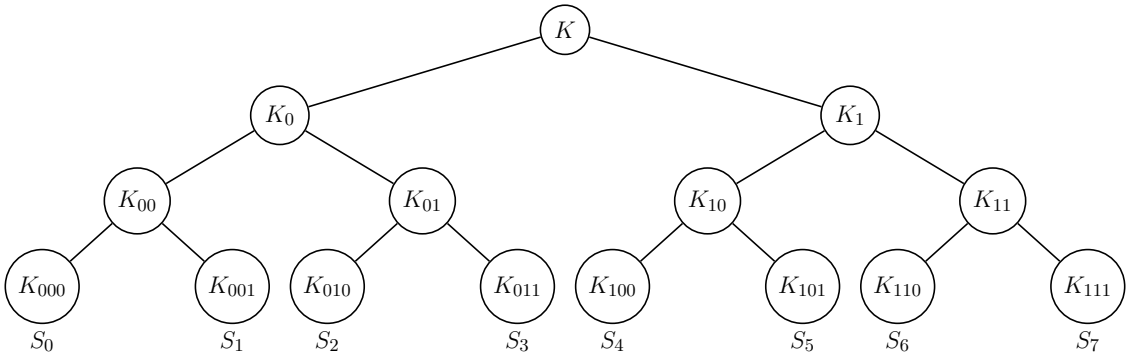

Figure 3.2: Balanced binary tree with eight leaves.

Structure. LKH uses a binary tree structure with the group key at the root and subscribers as leaf nodes. Each subscriber holds the keys along the path from its leaf to the root. The LKH scheme can be extended to k-ary trees, but this may reduce transmission efficiency. An example of this is shown in Figure 3.2. With 8 subscribers sharing the same topic, the broker generates a group key, K, to multicast the message to the subscribers. During initialization the broker creates a binary tree and provides the subscribers all the keys from the leaf to root. For example, subscriber S_1 will receive the following set of keys : $(K_{001}, K_{00}, K_0, K)$.

Revocation. To revoke a user at leaf x, the broker first simply updates all the

keys in the path from x to the root. To update the other users about this key change, the broker simply does the following - for all $v \in \texttt{Path}(v)$

- If v is a parent of x, then $K(v)$ is encrypted using $K(s(v))$, where $s(v)$ is the sibling of v
- Else, $K(v)$ is encrypted using both of its children node keys

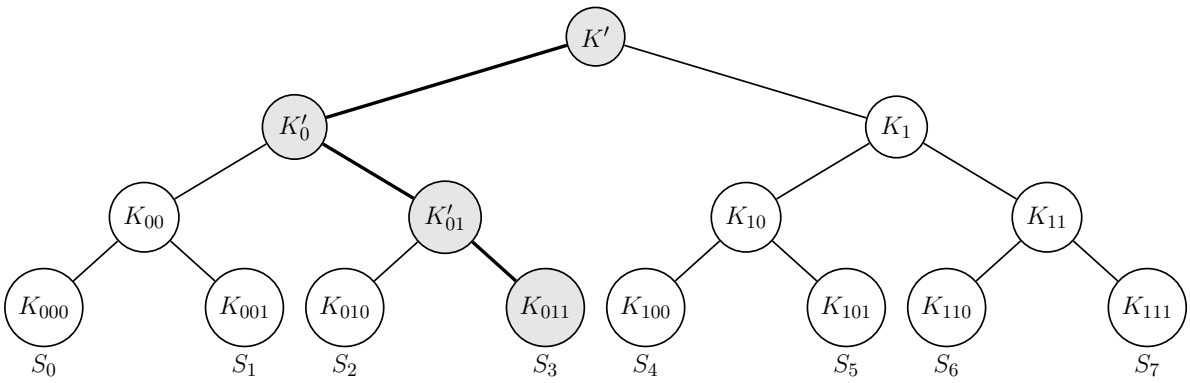

Figure 3.3: Revocation of S_3 using LKH scheme

For example, in Figure 3.3, to revoke the user S_3, the broker will

1. update the keys in $\texttt{Path}(S_3)$, $K_{01} \rightarrow K'_{01}$, $K_0 \rightarrow K'_0$, and finally the root $K \rightarrow K'$
2. broadcast $\{\texttt{Enc}\,(K'_{01}, K_{010})\}$, $\{\texttt{Enc}\,(K'_0, K'_{01}), \texttt{Enc}\,(K'_0, K'_{00})\}$, and $\{\texttt{Enc}\,(K', K'_0), \texttt{Enc}\,(K', K_1)\}$

All the nodes under K_1 will be able to receive the new group key K' by decrypting the broadcast message $\texttt{Enc}\,(K', K_1)$ since they are in possession of the key K_1. Similary, S_2 will first use the broadcast message $\texttt{Enc}\,(K'_{01}, K_{010})$ to retrieve K'_{01}. Using K'_{01}, it will be able to decrypt $\texttt{Enc}\,(K'_0, K_{01})$ and retrieve K'_0. Using K'_0, it will

finally get K' by decrypting $\texttt{Enc}(K', K'_0)$. It can be observed that the revoked user will not be able to decrypt any of these broadcast messages as it is not in possession of other keys. Collusion with non-revoked users is not interesting and is trivial. If it can simply collude with a non-revoked user it can simply get the data from the user. This method is secure against collusion of multiple revoked users, making it ideal for key management in my protocol.

Offine Key Management. Suppose a subscriber is offline for t re-keying operations due to either bandwidth issues or simply because of a business decision. The broker needs to update the subscriber with previous group keys and also update its state, i.e. change keys in its path. This can be done in $\mathcal{O}(t \log N)$ updates trivially using LKH. However, offline key updates were optimized by Pinkas [Pin02] to $\mathcal{O}(\log t)$ updates. A balanced binary tree is constructed using a pseudorandom generator (PRG) based tree construction method, where the total number of updates (additions or revocations) is predetermined [Pin02]. The group keys are positioned as leaves in this tree. By providing users with only $\mathcal{O}(\log t)$ values of these nodes, they can independently identify the t consecutive leaves by repeatedly applying the PRG. However, the security consequences of using this updated scheme is not investigated in this thesis. Simply using the LKH scheme itself is efficient for our purpose.

The method for key management in broadcast encryption has been extensively studied and there are several possible improvements over the LKH scheme. For example, Sherman et al. [SM03] uses One-way Function Trees (OFT), which halves the broadcast size of LKH while also providing the nodes to contribute entropy to the keys. Goodrich et al. [GST04] provide key management for low power devices, and particularly for zero state devices for which key updates are not at all possible, using

the Stratified Subset Difference (SSD) method. This technique can prove very useful in the scenarios where the devices cannot participate in key pairing. For now this solution has not been incorporated in the protocol, but is included in Future Work. LKH can also be used to add users in the system. The protocol works in exactly the same manner.

4 Protocol

The proposed protocol in the publish/subscribe messaging system utilizes two keys to encrypt messages, ensuring decoupling between subscribers and publishers. The broker broadcasts messages to topic subscribers, while the token manager provides tokens as the second key for decryption. This protocol is referred to as Two Key Broadcast Encryption (TKBE).

4.1 High Level Overview

The architecture consists of publisher, broker, subscriber and a Token manager. Before the protocol begins, we assume the following:

1. There is a Central/Decentralized Authority that publishes a global revocation list publicly (can be on a public blockchain)
2. she is able to verify credentials of each entity involved in the protocol
3. The ledger contains public keys of all the entities involved in this protocol

In a version where the Token Manager is completely trusted, we can assume that the Token Manager itself is the authority, however we will avoid that for now. Since this thesis is about creating a protocol/framework at application layer for publish/subscribe, deciding on the authorization framework is out of scope of this thesis. More about centralized or decentralized authorization frameworks can be found in [And+18; And+19]. As shown in Figure 4.1, the publisher and Token Manager

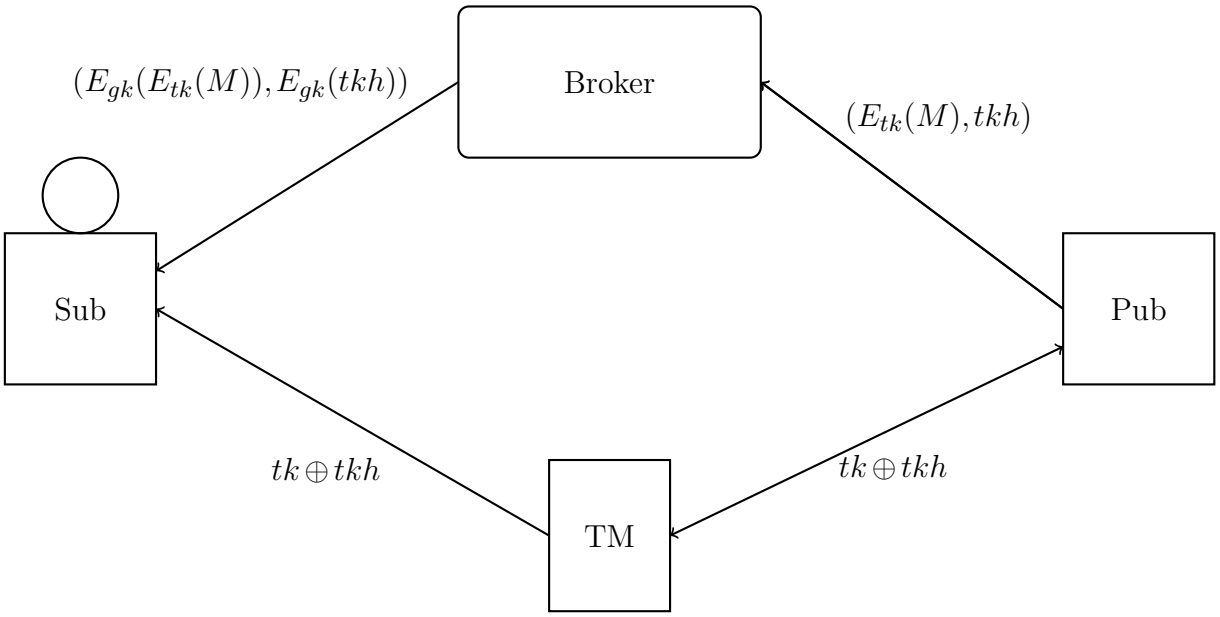

Figure 4.1: High Level Overview

(TM) establish a connection to agree on a token. The publisher encrypts and signs the message with a key and the key with a token. The Broker broadcasts the message and the encrypted key to the subscribers. The subscribers receive valid tokens from the Token Manager. Using the group key, the subscribers decrypt the message received from the Broker and decrypt the encrypted key using the valid token. Tokens are asynchronously refreshed between the Token Manager and Publisher during additions or revocations. Cryptographic hash functions and timestamps prevent

individual subscriber to refresh a token, because to refresh a token, the subscriber would need access to the random salt R_{i+1}, which she can only obtain if she has access to R_i. Subscribers require both the group key and a token from the Token Manager for message decryption following revocation or addition events. We use LKH scheme discussed earlier in Section 3 from the broker side to revoke a user.

4.2 Setup

Parameters. Public parameters are set by all the entities. We assume that the public keys of each entity participating in the protocol is available on the ledger. The elliptic curve domain parameters are set for the system.

Let $E(\mathbb{F}_q)$ be an elliptic curve with $q \in \mathbb{R}$ being the modulus for the Schnorr signature scheme, G being the generator. All the entities will generate $K = k \in^R \mathbb{Z}_q$ as secret key and kG as public key. Broker (k_B, K_B), Publisher (k_P, K_P), Subscriber (k_S, K_S) and Token manager (k_T, K_T) will register their public keys with the ledger. We are assuming only one subscriber and publisher for now, but this can be generalized by indexing.

KeyGen. The publisher generates $tk_0 \in^R \{0,1\}^\lambda$, where λ is the security parameter that will be used for symmetric encryption using `AES-GCM-SIV`. Here, the 0 in the subscript represents the time instance. tk_0 becomes tk_1 when a user is revoked/added for the topic corresponding to publisher. The broker will have a collection of random keys $gk_i \in^R \{0,1\}^\lambda$. These keys will serve as group keys for different topics.

4.3 Topic Registration

Publisher - Broker. The publisher generates $r \in^R \mathbb{Z}_q$ and computes $R = rG$, $Q = (Q_x, Q_y) = rK_b$. Let $s = Q_x$ be the shared secret between the publisher and the broker. The secret key is computed as $sk_{BP} = KDF(s)$, where $KDF(.)$ is a key derivation function [PM16]. Now the publisher computes the signature $sig = (z, e)$, where $e = H(topic, time, K_P, G, R)$ and $z = r - e.k_P$, where $H(.)$ is a hash function. The publisher then encrypts $(topic, time, K_P)$ using the secret key sk_{BP} in `AES-GCM-SIV` mode and obtains CT. The publisher then sends (CT, R, sig) to the Broker.

The Broker computes the shared secret by taking the x-coordinate of $k_B R$. She computes the shared secret key to decrypt the CT and checks the signature by computing $R' = zG + eR$. she computes $e' = H(topic, time, K_p, G, R')$, if $e = e'$ she accepts otherwise rejects the request. She then checks with the ledger if publisher is allowed to publish for the topic. Once verified, a session is formed between the publisher and broker. This similar process will be used again and again.

Subscriber - Broker. The subscriber registers its interest with the broker with the exact same process as above.

4.4 Key Distribution

Broker - Subscriber. For a topic, once the subscribers are registered, the broker creates a tree of $2n-1$ nodes rooted at gk_i for $topic_i$. Using the root, the Broker creates $2n-2$ more keys by a KDF chain. The leaves of this tree are the subscribers that are subscribed to the topic. The broker sends (CT, R, sig) similar to the process above to the subscriber, but here instead of $(topic, time, K_B)$ as plaintext, the broker encrypts $(topic, time, K_B, \texttt{Path}(x))$ as plaintext, where x is the node of the subscriber. The subscriber receives set of LKH keys.

Publisher - Token Manager. The publisher and token manager interact only once throughout the protocol. The publisher first generates $V_0 \in^R \{0,1\}^\lambda$ randomly and shares $PT = (V_0, time_0)$ with the token manager. The counter updates whenever a user is revoked/added for the topic that the publisher is publishing. The token manager collects such values for all the different topics. This message is also transmitted in the same way as above.

4.5 Key Synchronization

Publisher-Token Manager. From the previous step, it can be seen that the Token manager and the publisher have a shared random salt. It is assumed that the publisher and token manager are synchronized. Now, whenever a user is revoked/added, both the publisher and token manager compute $V_i = H(V_{i-1})$. Using V_i, the token manager then computes $H(V_i, time_i)$.

4.6 Publish

Publisher - Broker. Whenever a message is published by the publisher after n additions/revocations, the publisher first encrypts the message as $C = \texttt{Enc}(M, tk_n)$, then she encrypts $tkh_n = tk_n \oplus H(V_n, time_n)$. So the plaintext to be transmitted becomes $PT = (C, tkh_n)$. Then using the process above, PT is encrypted using the shared secret key of publisher and broker. The broker then receives the message, checks for the signature and if it is valid, she encrypts (C, tkh_n) using the group key gk_n and broadcasts it to the subscribers. It can be noted that without access to $H(V_n, time_n)$, the broker cannot decrypt tkh_n to obtain tk_n without which she cannot decrypt the message M.

4.7 Decrypt

Token Manager. The subscriber interacts with the token manager to receive the tokens. Using the encryption process described earlier, she sends the request as $(topic, time_i, K_S)$ to the token manager. The token manager verifies whether the subscriber is eligible to receive the token for that time frame through the ledger. If the subscriber is a valid subscriber, she receives $token_i = H(V_i, time_i)$. The token stays valid until another user is revoked or added.

Broker. If the subscriber is not revoked yet, she has still access to the group key. She can use the group key gk_i to retrieve (C, tkh_i). She can use $token_i$ to

obtain $tk_i = token_i \oplus tkh_i$ and then decrypt C using tk_i to obtain M. Since, we are assuming the broker is honest-but-curious we can be assured of the data integrity since the broker has already verified the data integrity from the publisher side. And the channel between subscriber and broker is authenticated and encrypted, so the subscriber does not need to check the signature of the publisher which retains the *decoupled* nature of communication.

4.8 Revocation[1]

Broker. When the ledger calls for revocation of a user, the broker broadcasts new encrypted keys and only the eligible users are able to decrypt the new group key. This time, rather than encrypting using individual keys for AES, the broker encrypts using the tree-keys and not the shared secret key. The LKH scheme mentioned in the previous section is used for revocation. To refresh the group key, the broker uses KDF to get the new key as $gk_n = KDF(gk_{n-1})$

.

Token Manager. The token manager after revocation, computes $V_n = H(V_{n-1})$. All the subscribers are automatically revoked this way. There is no way of obtaining future keys unless an adversary gets access to V_i at some point. Then she can compute the future keys.

Publisher. The publisher computes $V_n = H(V_{n-1})$ and she also computes $tk_n = KDF(tk_{n-1})$.

[1]Exactly the same algorithms are invoked on addition of a user

5 Conclusion

5.1 Assumptions

There are certain assumptions in this protocol.

- Non-collusion of broker and token manager
- Broker is honest but curious
- Authorization framework and ledger cannot be attacked
- The clocks are synchronized by clocks

5.2 Security

Based on the assumptions above, we achieve the following security guarantees.

- **Forward Secrecy**. If a user leaves the network, it won't be able to read the future messages, even if it colludes with either of broker or token manager because of the double revocation mechanism in place.

- **Backward Secrecy**. Any new user that joins the protocol will not have access to previous messages because of the one-way property of hash functions due to which the user won't have access to previous tokens even if she has access to a current token. Also, if she learns the current group key, she won't be able to learn the previous group keys [PM16].

- **Protection against replay attacks**. Since there are timestamps with every message, the attacker cannot replay a message.

- **Protection against revoked users colluding**. Even if the users collude, they cannot get the group key which was discussed in Chapter 3.

- **Protection against revoked users colluding with broker**. Even if the revoked users collude with the broker, they won't have access to tokens unless they attack the honest users. But if we assume that the revoked users collude with the honest users, the problem won't be interesting as they can directly obtain data from the users.

- **Protection against revoked users colluding with token manager**. If the users collude with the Token Manager we still have another layer of protection from the broker. The group key is not available to the revoked users, so they cannot decrypt the message.

- **Data Authentication and non-repudiation**. Data integrity and non-repudiation are ensure because of the Schnorr signature scheme.

- **Data privacy**. The data is encrypted using two keys and only the subscribers will be able to decrypt the message, unless the token manager and broker collude.

- **Decoupled Revocation**. The decoupling is maintained in this approach. The publisher would not need to change the key according to a particular revoked subscriber and thus without knowing the existence of the revoked user, the revocation is possible unlike some of the IBE based approaches.

5.3 Comparison

Table 5.1: Comparison of Protocols

Protocol	**Central Trust?**	**Decoupled Revocation**	**Data Privacy**	**Resource Constrained Devices**
JEDI [Kum+19]	No	No	Yes	Reduces battery life
Pub/Sub [Mal+19]	Yes	Yes, but uses revocation authority	Not from Broker	Lightweight
PICADOR [Bor+17]	Yes	No	From the broker, not from TTP	Similar to JEDI in performance
TKBE	Partial	Yes	Yes, under non-collusion	Lightweight

5.4 Parameters

The Schnorr signature scheme, `AES-GCM-SIV` and `ECIES` cryptosystem used in TKBE were taken from [Mal+19]. `AES-GCM-SIV` was used for its efficiency and less computational overhead, making it ideal for resource constrained devices. The security parameter λ and the values for q and elliptic curve can be decided by NIST standards.

6 Future Work

Authorization frameworks were not discussed in this thesis. However, there are frameworks that are decentralized and support Delegation of Trust [And+18; And+19]. More work can be done in this direction, to make the protocol more compatible with authorization frameworks like WAVE.

There are other approaches where one can use dynamic accumulators for set membership. The publisher can hold a single accumulator value and the subscriber proves in zero knowledge that it is a member of the set. Baldimtsi et al. [Bal+17] discusses one such approach with anonymous revocation. The set in this case would be of subscribers who are interested in a common topic. Binding ephemeral public keys with non-interactive proofs can provide a way to transmit data without central trust and have more anonymity.

The overhead of TKBE is expected to be comparable to [Mal+19] in theory. However, the primary goals include implementing and benchmarking the protocol to validate its efficiency and security.

Bibliography

[AKN08] Michel Abdalla, E. Kiltz, and Gregory Neven. "Generalised key delegation for hierarchical identity-based encryption". In: *Information Security, IET* 2 (Oct. 2008), pp. 67–78. DOI: 10.1049/iet-ifs:20070124.

[Ahm+17] Mian Ahmad Jan, Fazlullah Khan, Muhammad Alam, and Muhammad Usman. "A payload-based mutual authentication scheme for Internet of Things". In: *Future Generation Computer Systems* 92 (Sept. 2017). DOI: 10.1016/j.future.2017.08.035.

[And+19] Michael P Andersen, Sam Kumar, Moustafa AbdelBaky, Gabe Fierro, John Kolb, Hyung-Sin Kim, David E. Culler, and Raluca Ada Popa. "WAVE: A Decentralized Authorization Framework with Transitive Delegation". In: *28th USENIX Security Symposium (USENIX Security 19)*. Santa Clara, CA: USENIX Association, Aug. 2019, pp. 1375–1392. ISBN: 978-1-939133-06-9. URL: https://www.usenix.org/conference/usenixsecurity19/presentation/andersen.

[And+18] Michael P. Andersen, John Kolb, Kaifei Chen, Gabe Fierro, David E. Culler, and Randy Katz. "Democratizing Authority in the Built Environment". In: *ACM Trans. Sen. Netw.* 14.3–4 (Dec. 2018). ISSN: 1550-

4859. DOI: 10.1145/3199665. URL: https://doi.org/10.1145/3199665.

[Bal+17] Foteini Baldimtsi, Jan Camenisch, Maria Dubovitskaya, Anna Lysyanskaya, Leonid Reyzin, Kai Samelin, and Sophia Yakoubov. *Accumulators with Applications to Anonymity-Preserving Revocation*. Report Number: 043. 2017. URL: https://eprint.iacr.org/2017/043 (visited on 06/29/2023).

[BSW07] John Bethencourt, Amit Sahai, and Brent Waters. "Ciphertext-Policy Attribute-Based Encryption". In: *2007 IEEE Symposium on Security and Privacy (SP '07)*. 2007, pp. 321–334. DOI: 10.1109/SP.2007.11.

[BS20] Dan Boneh and Victor Shoup. "A graduate course in applied cryptography". In: *Draft 0.5* (2020). URL: http://toc.cryptobook.us/.

[Bor+17] Cristian Borcea, Arnab "Bobby" Deb Gupta, Yuriy Polyakov, Kurt Rohloff, and Gerard Ryan. "PICADOR: End-to-end encrypted Publish–Subscribe information distribution with proxy re-encryption". en. In: *Future Generation Computer Systems* 71 (June 2017), pp. 177–191. ISSN: 0167-739X. DOI: 10.1016/j.future.2016.10.013. URL: https://www.sciencedirect.com/science/article/pii/S0167739X16303983 (visited on 06/26/2023).

[Bra+21] Lorenzo Bracciale, Pierpaolo Loreti, Emanuele Raso, and Giuseppe Bianchi. "TooLate: Cryptographic Data Access Control for Offline Devices through Efficient Key Rotation". In: *Proceedings of the 2th Workshop on CPS&IoT Security and Privacy*. CPSIoTSec '21. Virtual Event, Republic of Korea: Association for Computing Machinery, 2021, pp. 57–62. ISBN: 9781450384872.

DOI: 10.1145/3462633.3483982. URL: https://doi.org/10.1145/3462633.3483982.

[BS18] Robert Bryce and Gautam Srivastava. "The Addition of Geolocation to Sensor Networks". In: Jan. 2018, pp. 796–802. DOI: 10.5220/0006921907960802.

[Fou20] Sovrin Foundation. *Self-Sovereign Identity and IoT*. en. Aug. 2020. URL: https://sovrin.org/library-iot/.

[GST04] Michael Goodrich, Jonathan Sun, and Roberto Tamassia. "Efficient Tree-Based Revocation in Groups of Low-State Devices". In: Aug. 2004, pp. 511–527. ISBN: 978-3-540-22668-0. DOI: 10.1007/978-3-540-28628-8_31.

[Goy+06] Vipul Goyal, Omkant Pandey, Amit Sahai, and Brent Waters. "Attribute-Based Encryption for Fine-Grained Access Control of Encrypted Data". In: *Proceedings of the 13th ACM Conference on Computer and Communications Security*. CCS '06. Alexandria, Virginia, USA: Association for Computing Machinery, 2006, pp. 89–98. ISBN: 1595935185. DOI: 10.1145/1180405.1180418. URL: https://doi.org/10.1145/1180405.1180418.

[GLL19] Shay Gueron, Adam Langley, and Yehuda Lindell. *AES-GCM-SIV: Nonce Misuse-Resistant Authenticated Encryption*. RFC 8452. Apr. 2019. DOI: 10.17487/RFC8452. URL: https://www.rfc-editor.org/info/rfc8452.

[Kar+15] Vasileios Karagiannis, Periklis Chatzimisios, Francisco Vazquez-Gallego, and Jesus Alonso-Zarate. "A survey on application layer protocols for

the internet of things". In: *Transaction on IoT and Cloud computing* 3.1 (2015), pp. 11–17.

[Kat+17] Sotirios Katsikeas, Konstantinos Fysarakis, Andreas Miaoudakis, Amaury Van Bemten, Ioannis Askoxylakis, Ioannis Papaefstathiou, and Anargyros Plemenos. "Lightweight & secure industrial IoT communications via the MQ telemetry transport protocol". In: *2017 IEEE Symposium on Computers and Communications (ISCC)*. 2017, pp. 1193–1200. DOI: 10.1109/ISCC.2017.8024687.

[Kha+22] Abid Khan, Awais Ahmad, Mansoor Ahmed, Jadran Sessa, and Marco Anisetti. "Authorization schemes for internet of things: requirements, weaknesses, future challenges and trends". en. In: *Complex & Intelligent Systems* 8.5 (Oct. 2022), pp. 3919–3941. ISSN: 2198-6053. DOI: 10.1007/s40747-022-00765-y. URL: https://doi.org/10.1007/s40747-022-00765-y (visited on 06/29/2023).

[Kum+19] Sam Kumar, Yuncong Hu, Michael P. Andersen, Raluca Ada Popa, and David E. Culler. "JEDI: Many-to-Many End-to-End Encryption and Key Delegation for IoT". In: *CoRR* abs/1905.13369 (2019). arXiv: 1905.13369. URL: http://arxiv.org/abs/1905.13369.

[Lak+20] Kamlesh Lakhwani, Hemant Kumar Gianey, Joseph Kofi Wireko, and Kamal Kant Hiran. *Internet of Things (IoT) : Principles, Paradigms and Applications of IoT*. BPB Publications, 2020.

[LTP22] Apostolos Lazidis, Konstantinos Tsakos, and Euripides G. M. Petrakis. "Publish–Subscribe approaches for the IoT and the cloud: Functional and performance evaluation of open-source systems". en. In: *Internet of Things* 19 (Aug. 2022). DOI: 10.1016/j.iot.2022.100538.

[Mal+16] Lukas Malina, Jan Hajny, Radek Fujdiak, and Jiri Hosek. "On Perspective of Security and Privacy-Preserving Solutions in the Internet of Things". In: *Computer Networks* 102 (Mar. 2016). DOI: 10.1016/j.comnet.2016.03.011.

[Mal+19] Lukas Malina, Gautam Srivastava, Petr Dzurenda, Jan Hajny, and Radek Fujdiak. "A Secure Publish/Subscribe Protocol for Internet of Things". In: ARES '19. Canterbury, CA, United Kingdom: Association for Computing Machinery, 2019. DOI: 10.1145/3339252.3340503. URL: https://doi.org/10.1145/3339252.3340503.

[MQT] MQTT. *MQTT : The standard for IoT Messaging*. URL: https://mqtt.org/.

[PM16] Trevor Perrin and Moxie Marlinspike. "The double ratchet algorithm". In: *GitHub wiki* (2016), p. 112.

[Pin02] Benny Pinkas. "Efficient State Updates for Key Management". In: *Security and Privacy in Digital Rights Management*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2002, pp. 40–56. ISBN: 978-3-540-47870-6.

[Pol+17] Yuriy Polyakov, Kurt Rohloff, Gyana Sahu, and Vinod Vaikuntanthan. *Fast Proxy Re-Encryption for Publish/Subscribe Systems*. Report Number: 410. 2017. URL: https://eprint.iacr.org/2017/410 (visited on 07/04/2023).

[Sch90] C. P. Schnorr. "Efficient Identification and Signatures for Smart Cards". In: *Advances in Cryptology — CRYPTO' 89 Proceedings*. Ed. by Gilles Brassard. New York, NY: Springer New York, 1990, pp. 239–252. ISBN: 978-0-387-34805-6.

[SM03] A.T. Sherman and D.A. McGrew. “Key establishment in large dynamic groups using one-way function trees”. In: *IEEE Transactions on Software Engineering* 29.5 (2003), pp. 444–458. DOI: 10.1109/TSE.2003.1199073.

[Sho01] Victor Shoup. *A Proposal for an ISO Standard for Public Key Encryption*. Cryptology ePrint Archive, Paper 2001/112. https://eprint.iacr.org/2001/112. 2001. URL: https://eprint.iacr.org/2001/112.

[WHA99] D. Wallner, E. Harder, and R. Agee. *RFC2627: Key Management for Multicast: Issues and Architectures*. USA, 1999.

[Wan+16] Frank Wang, James Mickens, Nickolai Zeldovich, and Vinod Vaikuntanathan. “Sieve: Cryptographically Enforced Access Control for User Data in Untrusted Clouds”. In: *13th USENIX Symposium on Networked Systems Design and Implementation (NSDI 16)*. Santa Clara, CA: USENIX Association, Mar. 2016, pp. 611–626. ISBN: 978-1-931971-29-4. URL: https://www.usenix.org/conference/nsdi16/technical-sessions/presentation/wang-frank.

[Xu+18] Hansong Xu, Wei Yu, David Griffith, and Nada Golmie. “A survey on industrial Internet of Things: A cyber-physical systems perspective”. In: *Ieee access* 6 (2018), pp. 78238–78259.

[ZME21] Lin Zhu, Suryadipta Majumdar, and Chinwe Ekenna. “An invisible warfare with the internet of battlefield things: A literature review”. In: *Human Behavior and Emerging Technologies* 3.2 (2021), pp. 255–260. DOI: https://doi.org/10.1002/hbe2.231. eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1002/hbe2.231.