T-HIBE: A Novel Key Establishment Solution for Decentralized, Multi-Tenant IoT Systems

Sayon Duttagupta, Dave Singelée, Bart Preneel
KU Leuven - imec - COSIC, Belgium
Email: firstname.lastname@esat.kuleuven.be

***Abstract*—The Internet of Things (IoT) devices has evolved considerably in the past few years and is expected to grow exponentially in the next decade. This exponential growth makes key management in an IoT ecosystem very challenging. Traditional IoT systems are often centralized and grouped into an ecosystem. However, this type of centralized architecture is not always compatible with practical IoT deployments. This paper proposes T-HIBE, a secure key establishment and agreement solution for a decentralized multi-tenant IoT system with multiple security domains. T-HIBE relies on principles of identity-based cryptography for key transport between intra and inter-domain devices while avoiding the inherent key-escrow problem. Furthermore, we have demonstrated our proposed architecture on an ARM Cortex-M4 microcontroller and evaluated the performance to show that T-HIBE does not have a significant energy and performance cost.**

I. Introduction

There is an abundance of Internet-of-Things (IoT) devices today, and the number of devices is rising exponentially. It is now being used in various domains, from smart homes, healthcare and environmental sensors to large scale industries and supply-chain management. There is a growing need to establish secure connections between these devices to exchange data while preserving confidentiality and integrity. Key management becomes a challenge in these domains, with the ever-increasing number of devices. Traditionally, securing devices under the same security domain is typically done via symmetric-key encryption. Each IoT device will have a pre-shared key installed during fabrication and can use this to communicate with a trusted third party (TTP), which acts as a key distribution server in the network. When two IoT devices in the network want to establish a session key, they rely on the TTP to generate and securely transport a fresh session key to both IoT devices. However, much trust is involved in the TTP, as it is the central point of failure in the system. Suppose one wants to extend this concept to a multi-security domain setting, where two IoT devices from different security domains want to establish a secure connection. First, the TTPs of the security domains involved would need to agree on a session key. Next, each TTP then securely transports the session key to the IoT device within its security domain. The disadvantage of this approach is that both TTPs learn the session key, so the IoT device in the first security domain would also have to trust the TTP in the second security domain. Moreover, each time a session key needs to be established, an online connection to the TTP involved is required. These solutions put a lot of trust in these centralized hubs and act as a single point of failure. Moreover, this solution does not scale very well with an increasing number of IoT devices. Key management becomes exponentially large, and more throughput is exerted on the central hub. An alternative solution that is quite popular is to use SSL/TLS for all the communications and connect it to the main router or switch. This increases the load on such networks and does not scale very well, especially in large IoT ecosystems such as industrial ones. Furthermore, with the increasing number of devices, certificate management also becomes increasingly difficult.

An important trend is an evolution towards decentralized IoT ecosystems, where IoT devices from multiple security domains can interact and exchange data. A decentralized IoT system provides a more feasible solution for scalability and interoperability, and single-point-of-failure problems. Moreover, one can expect that soon, more IoT devices in such decentralized ecosystems will have the ability to initiate transactions among themselves. This will significantly reduce the execution time and communication costs, as they can communicate among themselves and not rely on a central TTP. Despite the advantages of symmetric-key cryptography – being computationally inexpensive compared to their asymmetric counterparts and having a smaller key length, it is clear that one needs public-key cryptography to realize key establishment in a decentralized IoT ecosystem. Generally, this can be done using digital certificates and a robust public-key infrastructure (PKI). Usually, this involves a central authority, known as the Certificate Authority (CA), that issues digital certificates of a device's public-key. However, deploying a large-scale PKI for a decentralized IoT system with multiple security domains is challenging. This would require the CA to issue certificates to each IoT device, which is not realistic. Moreover, a disadvantage of deploying a PKI for IoT is that revocation can become challenging. Before any key between two IoT devices can be established, these devices would need to check that the other party's certificate is still valid. This requires either a connection to an online service or the regular distribution of Certificate Revocation Lists. Both approaches are rather impractical for a decentralized IoT setting. Therefore, in our work, we present an alternative approach based on Identity-Based Encryption (IBE). More specifically, we propose T-HIBE, a novel key establishment scheme for multi-tenant IoT security systems. This decentralized methodology of key establishment in IoT devices facilitates multi-domain

key exchanges. Although our solution can be realized in any IoT system where a decentralized and heterogeneous solution is needed, in the rest of the paper, we put forward the example where each security domain is managed by a user (tenant). We envision our work in a smart apartment setting, with multiple tenants as entities or users, and each of them having multiple IoT devices which can interact with each other. These devices should be able to establish secure connections with IoT devices from other domains.

II. Cryptographic Background

Before introducing our novel security solution, we will first discuss some preliminaries on which our cryptographic schemes are built.

A. Bi-linear pairings

Pairings are an additional structure-property demonstrated by some curves that give way to a branch of public-key cryptography known as Pairing-based Cryptography (PBC). A pairing $\hat{e}$ abstractly operates on two groups - a source group $\mathbb{G}$ and a target group $\mathbb{G}_T$. Normally the source group will be points on an elliptic curve, and the target group be elements in a finite field. A pairing takes two points in the source group and maps them to the target group so that the exponents multiply. Hence we say the pairings is bilinear, and by bilinear, we mean the multiplication of exponents. Formally, a bilinear pairing is a map, which can be defined as follows:

Let n be a large prime number. Let $\mathbb{G}_1$ and $\mathbb{G}_T$ be two cyclic groups of prime order n, where $\mathbb{G}_1$ is represented additively with identity ∞ and $\mathbb{G}_T$ is represented multiplicatively with identity 1. The map $\hat{e}$ is represented as,

$$\hat{e} : \mathbb{G}_1 \times \mathbb{G}_1 \rightarrow \mathbb{G}_T$$

that will have the following conditions:

1) **Bilinear:** $\hat{e}(P^a, Q^b) = \hat{e}(P^b, Q^a) = \hat{e}(P, Q)^{ab}, \quad \forall a,b \in \mathbb{Z},\ P,Q \in \mathbb{G}_1$
2) **Computable**: There is an efficient algorithm to compute $\hat{e}$ in polynomial-time.
3) **Non-Degenerate:** $\hat{e}(P, P) \neq 1$, for some $P \in \mathbb{G}_1$

There are some consequences of pairings in the standard assumption models. The **Decision Diffie-Hellman** (DDH) problem in $\mathbb{G}_1$ is easy [14], that is, if for a given input g, g^x, g^y, $g^z \in \mathbb{G}_1$ to test whether $z = xy$ is as simple as computing $\hat{e}(g, g^z) = \hat{e}(g^x, g^y)$. Hence it is not possible to do standard Diffie-Hellman protocol in pairing groups. Also, it is possible to efficiently reduce a discrete-log problem in $\mathbb{G}_1$ to a discrete-log problem in the target group $\mathbb{G}_T$ [17]. Hence it is imperative that the discrete-log in the target group $\mathbb{G}_T$ is hard, or else the discrete-log in the source group $\mathbb{G}_1$ will not be hard. The security of the bilinear property depends on the Discrete-Log Problem (DLP) to be hard, and the Computational Diffie-Hellman (CDH) problem to be hard as well. The Bilinear Diffie-Hellman (BDH) problem is defined as, for a randomly chosen $P \in \mathbb{G}_1$, as well as aP, bP and cP, for some randomly chosen $a,b,c \in \mathbb{Z}$, it is computationally hard to compute $\hat{e}(P, P)^{abc}$. Also, for the BDH problem to be hard, $\mathbb{G}_1$ and $\mathbb{G}_T$ must be chosen in a way so that no known algorithm can efficiently solve the Diffie-Hellman problem in either $\mathbb{G}_1$ or $\mathbb{G}_T$.

B. IBE

In contrast to traditional PKI models where one needs to have their public keys authenticated by a CA, in identity-based cryptography, or Identity-Based Encryption (IBE), the public key is essentially the public identity itself and does not need any further authentication. The idea of identity-based cryptography and its signature schemes was first envisioned by Shamir [20] in 1984, but it was formally realized independently by Boneh and Franklin [4] and Cocks [8] in 2001. The Boneh and Franklin model is based on the Bilinear Diffie-Hellman problem, whereas the Cocks scheme is based on the quadratic residuosity problem. The primary motivation for identity-based encryption was to help the deployment of a public key infrastructure [4].

The private-key generator (PKG) is the central trusted entity in the IBE scheme responsible for functioning the entire protocol. It acts as the key generator center, and runs the setup phase and key extraction phase, and then sends the private keys of the respective devices to themselves via a secure channel, as shown in Figure 1. A device's private key is derived from the master secret-key of the PKG and the device's ID itself, which is also the public-key. Therefore, as only the PKG knows the master secret-key, it can only derive the subsequent private-keys of the devices. Then the devices can communicate securely with each other, using the parameters received from the PKG and the devices' identities. For the setup phase in the Boneh-Franklin IBE scheme, the PKG will randomly generate a generator $P_0 \in \mathbb{G}_1$ and a private-key $x \in \mathbb{Z}$, and computes the public key as xP_0. During the key extraction phase, a device with identity ID will request the PKG to generate its private-key, and the PKG would do so by computing $xH(ID)$, where H is a cryptographic hash function. To encrypt a message

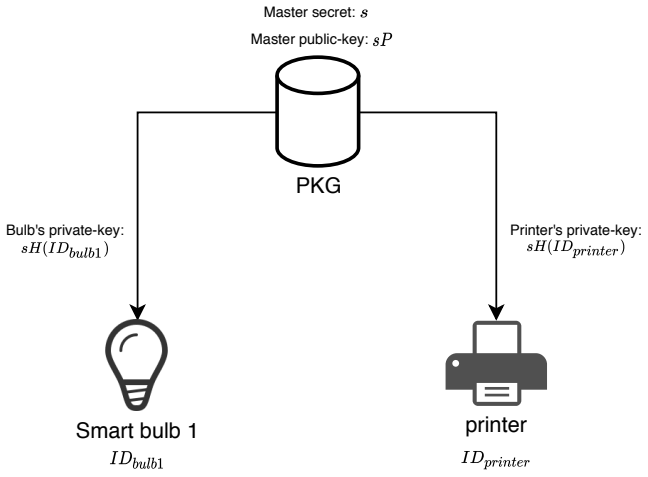

Fig. 1: Identity-based Encryption on an IoT system.

$\mathcal{M}$, only the identity ID and the master public-key xP_0 is required. The device can then decrypt it using its private-key.

C. HIBE

In a traditional IBE scheme, a single PKG generates the private keys for all the devices and transmits them via a secure channel. Subsequently, significant research effort has been devoted to realizing IBE and HIBE schemes. HIBE, or hierarchical IBE, is, in essence, a multi-layer IBE, where the private-key generation can be delegated to other devices in the hierarchical order. A HIBE will have multiple PKGs for the multiple levels, as depicted in 2. A layer-1 HIBE is identical to a normal IBE. It can be scaled further by making the root devices under the root PKG as the new PKGs for its subsequent devices and thus creates a hierarchical structure. HIBE is more scalable than a traditional IBE scheme, and it divides the task of private-key generation from one PKG to multiple PKGs, thus easing the burden on the root PKG. An advantage HIBE has over IBE schemes is that leakage of any domain level private key of the domain-specific PKG will not compromise the secrets of the higher level PKGs. The identities of the devices will be with respect to their PKGs and their hierarchical depth. More specifically, identities here are tuples; hence a device ID_d at level-3 will have an identity "$\mathtt{ID}_{\mathtt{root}}||\mathtt{ID}_{\mathtt{domain}}||\mathtt{ID}_{\mathtt{d}}$".

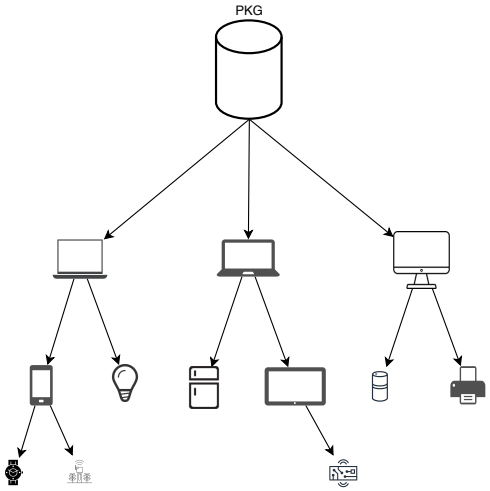

Fig. 2: Hierarchical IBE.

Horwitz and Lynn [13] first introduced the concept of HIBE with a formal definition of a 2-level HIBE model. The main goal was to reduce the throughput and workload of the PKG by delegating its work to other devices. There have been further constructions of hierarchical IBE since then. The Gentry and Silverberg construction [12] is IND-IDCPA secure in the RO model, under the BDH assumption. The Boyen-Waters scheme [5] is both anonymous and selectively secure without random oracles. However, in all of these constructions, the ciphertext grows linearly with hierarchical depth. Hence deeper the hierarchy, the bigger is the ciphertext. Also, security degrades exponentially in the hierarchy depth. Depth 2, 3 or 4 is considered fine, but more than that has a significant impact on security. The Boneh-Boyen-Goh construction [3] is a selectively secure construction where the ciphertext is independent of the hierarchy depth. However, the public parameter in this scheme is dependent on the depth of the hierarchy; hence, one has to commit to the depth of the hierarchy ahead of time, unlike the previous ones, which can be scalable further on the fly. HIBE, like IBE, suffers from key-escrow. Moreover, in HIBE especially, the PKG at level h can read all information of level $h+1, h+2, ...n$. Hence, the root PKG can read every encrypted data of every level.

III. Attacker Model and Assumptions

In an IoT environment with a private-key generator (PKG), we inherently assume that the PKG is *honest, but curious*. As in an IBE setting, the PKG is responsible for generating and distributing keys, it can also decrypt any communication between devices that has received a key-pair from the PKG. Furthermore, in a hierarchical setting, the PKG(s) at a higher level can read the information being exchanged at their lower levels. We assume that in the initial setup phase, only the key-pair distribution is done via a secure channel. This secure channel is only required for this phase. Once it is complete, any two devices in the scheme can agree on a key. All the public-keys, names and IDs of the devices are publicly known and available. The attacker is assumed to have access to the insecure wireless channel during the execution of the protocol and can intercept, inject or modify messages at will and try to break the confidentiality and read the messages.

IV. Our solution: Threshold-based Hierarchical Identity-Based Encryption (T-HIBE)

The discussion in the section above shows that using IBE in an IoT setting has multiple benefits. Compared to a standard PKI model, IBE offers less storage and memory overhead with a slight compromise on the execution time. There is no need to hold and store certificates and do an online lookup. The HIBE scheme discussed in Sect. II-C naturally fits the multi-security domain setting we envision in this paper. Two IoT devices could use it as a cryptographic primitive to establish a joint session-key, as long as the security domains of both IoT devices have a common ancestor in the hierarchical domain system.

However, a common problem with both IBE and HIBE is key-escrow. As the (root) private-key generator (PKG) is responsible for generating the private keys, it can, in theory, decrypt all the messages for all the devices. Although the PKG is assumed to be a trusted entity, this might be an unfavourable situation in some IoT use cases. Hence, we propose T-HIBE: A threshold-based hierarchical identity-based encryption.

A. Main concept

The basic idea of T-HIBE is to divide the functionality of the root private-key generator over multiple, non-colluding root PKGs. These different root PKGs only need to agree on public parameters, including a public key. Each of the root PKGs will generate its own local private master key. When a node

on the layer below needs to receive its private key, each of the root PKGs will compute a *share* of the private key by using their local master key. The node can then use Shamir's secret-sharing [19] on the different shares to compute its private-key[1].

When having computed its private key, each node on the layer below the root layer can, in turn, be the new PKG in his trusted security domain and generate and distribute keys to the IoT devices on the layer below, as shown in Figure 3. If all the root PKGs do not collude with each other, then no messages can be decrypted, as none of the root PKGs in the system know the actual master key (i.e. the master key that hypothetically would be the result of combining all the master keys of the different root PKGs). The domain PKG (i.e. the user in Figure 3) can, however, still do this, but as this is intra-domain, this is considered inside a trusted ecosystem or a trusted domain. For example, in a multi-tenant system, the different users can be the different tenants, and they, in turn, distribute the keys to their own local IoT devices.

B. Protocol structure

For our scheme, we are using Gentry-Silverberg construction of HIBE [12], which is known to be one-of-the only fully functioning HIBE up to date [9]. We use Pedersen's distributed key generation algorithm [18] in a verifiable (t, n) secret-sharing scheme to divide the shares of different root PKGs, such that there is no "honest-dealer" to distribute the shares, but each root PKG can compute and verify on their own.

To formally realize this, we have divided our scheme into multiple sections. **PKG initialization** is where all the different PKGs are initialized and setup, and the threshold secret-keys are generated and computes the master secret-key and agrees on a master public-key. For our implementation (Sect. VII), we have envisioned three honest-but-curious PKGs, but it can be practically deployed to any n numbers of PKGs, which requires at least a set of t keys to compute the final secret

[1]Note that this result would be equal to the hypothetical case where the different root PKGs would first combine their master-keys, using Shamir's secret-sharing, to compute a *joint root master-key*. And then compute the private-key of the node using this joint root master-key. Since the root PKGs are assumed not to collude, this hypothetical case will never happen, and none of the parties in the system will know the joint root master-key.

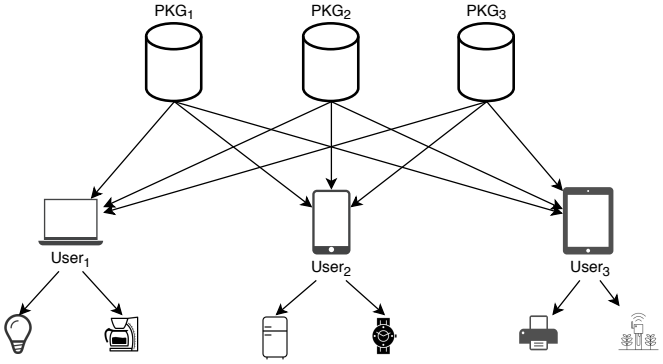

Fig. 3: T-HIBE on an IoT system.

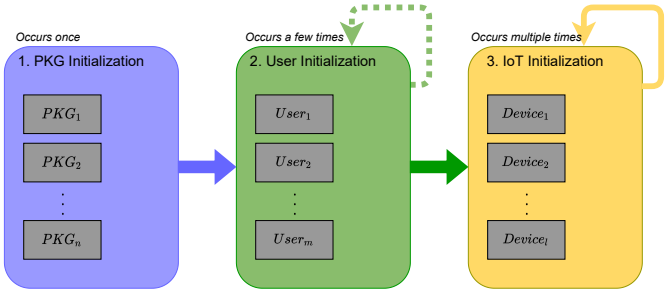

Fig. 4: Initialization phases of T-HIBE.

(Sect. VI-A). If required, the protocol also has the option to add more PKGs into the ecosystem after the system goes online. The PKG initialization phase needs to occur only once per protocol execution, and every time a new PKG needs to be set up. Ideally, these PKGs can be operated on the cloud and can be co-located at different places, but if need be, they can even be realized on a microcontroller (Sect. VII). This phase is followed by **User initialization**, where t minimum number of PKGs send their individual shares of the user's private-key, and the user extracts and computes the final private-key locally on their device. The users obtain their shares of their private-keys via secure external channels, such as TLS, and then combine these shares locally on their device via threshold cryptography to yield the final secret. These user devices, in turn, becomes the new PKG for their sub-domain. This phase can be re-run a few times to periodically refresh the sub-keys to avoid key leakages and maintain the security level. However, frequent execution of this level would lead to a less efficient protocol as all the devices under the user also need to re-run the protocol again to get their newly generated keys. Finally, **IoT device initialization** deals with IoT devices initializing in their domains and agreeing on a session-key. A new IoT device will obtain their private-key from their domain user via some secure channel. After all devices and phases have been initialized, **key-establishment** occurs, which will then create a symmetric session-key, and transport this session-key as an encrypted message via T-HIBE to any other device in the ecosystem and will use this specific session-key to encrypt data and communicate between IoT devices until revoked or timed out. This level can be executed multiple times to avoid the keys from long-term leakage. Theoretically, our T-HIBE scheme can be extended to multiple hierarchical levels also, but to avoid any loss in efficiency and security, as the ciphertext length grows linearly with more levels, we believe this structure of using three levels is apt.

V. Protocol Construction

A. PKG initialization - key generation

As defined by Gentry-Silverberg [12], this can be formally defined by the following steps:

1) Let us assume there are n PKGs: $P_1, P_2, P_3, ..., P_n$.
2) Only t out of the n PKGs are required to compute the secret.

3) Let any one of the PKG, P_i, run the BDH parameter generator [4] $\mathcal{IG}$ to output two groups $\mathbb{G}_1$ and $\mathbb{G}_2$ of the same prime order q and the description of a suitable pairing $\hat{e} : \mathbb{G}_1 \times \mathbb{G}_1 \to \mathbb{G}_2$.
4) Selects cryptographic hash functions $H_1 : \{0,1\}^* \in \mathbb{G}_1$ and $H_2 : \mathbb{G}_2 \in \{0,1\}^n$ for some n.
5) Selects a random generator $P \in \mathbb{G}_1$; P_i then broadcasts these public values to all the other PKGs.
6) Each PKG P_i then randomly selects a secret $s_i \in \mathbb{Z}_q$ and computes $Q_i'' = s_i P$ and $a_{i0} = s_i$.
7) Each PKG P_i selects a random polynomial $f_i(x) \in \mathbb{Z}_q$ of degree $t-1$ of the form:

$$f_i(x) = a_{i0} + a_{i1}x + a_{i2}x^2 + ... + a_{i(t-1)}x^{t-1} \quad (1)$$

where $f_i(0) = a_{i0} = s_i$

8) Every PKG P_i will compute and send $A_{ik} = a_{ik}P$ for $k \in [0, t-1]$ to all other PKGs, and $A_{i0} = s_i P = Q_i''$.
9) Every PKG P_i will compute the share $s_{ij} = f_i(j) \bmod q$ and send it to PKG P_j via a secure channel, for all $i, j \in [1, t]$
10) Each PKG P_i can now verify the all the $t-1$ shares received from the other PKGs by computing

$$s_{ij}P = \sum_{k=0}^{t} j^k A_{ik} \quad (2)$$

for all $i, j \in [1, t]$

11) After all the PKGs have received their share of s_{ij}, they can compute their respective shares. $F(i)$ is the share of PKG P_i, and is denoted as:

$$F(i) = f_1(i) + f_2(i) + f_3(i) + ... + f_t(i) = \sum s_{ij} \quad (3)$$

for $i, j \in [1, t]$, where $F(i)$ is the share of PKG P_i, and $F(0)$ is the master-secret key.

12) To compute the master-secret s, theoretically, one could compute the Lagrange's Interpolation on the polynomial, namely,

$$s = \sum_{i=0}^{t} L_i(0)F(i) \quad (4)$$

Where,

$$L_i(x) = \frac{\prod_{j=1}^{t} x - j}{\prod_{\substack{j=1 \\ j \neq i}}^{t} i - j} \quad (5)$$

13) The public point Q can be computed as,

$$Q = \sum_{i=0}^{t} L_i(0)Q_i'$$

Where,

$$Q_i' = F(i)P$$

14) Message space is $\mathcal{M} = \{0,1\}^m$ and the ciphertext space is $\mathcal{C} = \mathbb{G}_1^h \times \{0,1\}^n$, h being the hierarchical level of the recipient.
15) Public parameters are defined as $PK = (\mathbb{G}_1, \mathbb{G}_2, \hat{e}, P, Q, H_1, H_2)$

B. PKG initialization - key extraction

1) In HIBE, the identity of a user is defined by the tuple of identities till the root PKG, where the root PKG is the collection of all the threshold PKGs. Hence, the User 1 as shown on Fig 3 will have its identity as $ID_{u1} =$ $(\mathtt{ID_{root}} || \mathtt{ID_{u1}})$.
2) Every PKG P_i will independently compute the share $\gamma_i = F(i)H(ID_{u1})$ and send it to the user's device, over a secure channel, to compute the user's secret key $sH(ID_{u1})$ locally, in the following manner:

$$sH(ID_{u1}) = \sum_{i=0}^{t} L_i(0)\gamma_i$$

Where,

$$L_i(x) = \frac{\prod_{j=1}^{t} x - j}{\prod_{\substack{j=1 \\ j \neq i}}^{t} i - j}$$

3) None of the PKGs learn anything about the final secret of the domain member (user), unless t or more of them collude, which is highly unlikely and violates our initial security assumption. Leakage of $t-1$ shares will not leak any information about the final secret.

C. Users' and Devices initialization - key extraction

1) Identity is in the form of tuples, and User $H+1$ at level $h+1$ will have an ID-tuple of $(\mathtt{ID_1}, \ldots, \mathtt{ID_h}, \mathtt{ID_{h+1}})$, where $\mathtt{ID_1}$ is the root-PKG layer.
2) Every user at level $h \geq 0$ has a secret point $S_h \in \mathbb{G}_1$ and $(h-1)$ translation points $Q_1, ...Q_{h-1} \in \mathbb{G}_1$.
3) To compute the secret-key for the device $\mathtt{ID_{h+1}}$ at level $h+1$, the parent $\mathtt{ID_h}$ computes $P_{h+1} = H_1(\mathtt{ID_1}, \ldots, \mathtt{ID_h}, \mathtt{ID_{h+1}}) \in \mathbb{G}_1$ will pick a random secret $s_h \in \mathbb{Z}_q$ and sets the child's $\mathtt{ID_{h+1}}$ secret point as $S_{h+1} = S_h + s_h P_{h+1}$, and the translation point as $Q_h = s_h P$, and send all these values, along with the previous $h-1$ translation points to the child via a secure channel. Hence, the child's $\mathtt{ID_{h+1}}$ secret key is

$$S_{h+1} = \sum_{i=1}^{h+1} s_{i-1}P_i, Q_1 = s_1 P, ..., Q_h = s_h P \quad (6)$$

D. Message encryption

1) To encrypt a message $M \in \mathcal{M}$, for device $\mathtt{ID_h}$ at level h with ID-tuple $(\mathtt{ID_1}, \ldots, \mathtt{ID_h})$, compute $P_i = H(\mathtt{ID_1}, \ldots, \mathtt{ID_i})) \in \mathbb{G}_1 \forall 1 \leq i \leq h$ and choose a random $r \in \mathbb{Z}_q$.

2) The ciphertext is,

$$C = [rP, rP_2, ..., rP_h, M \oplus H_2(g^r)] \quad (7)$$

$$\text{where } g = \hat{e}(Q, P_1) \in \mathbb{G}_2$$

3) To decrypt the ciphertext, let $C = [U, U_2, ..., U_h, V] \in \mathcal{C}$ be the ciphertext.
4) The message will be,

$$M = V \oplus H_2(\frac{\hat{e}(U, S_h)}{\prod_{i=2}^{h} \hat{e}(Q_{i-1}, U_i)}) \quad (8)$$

E. Session-key establishment

1) As the ciphertext length grows linearly with hierarchical depth, and there might be a lot of message transmissions in quick successions in an IoT environment, it is simpler to establish a symmetric session-key and use this for the actual encryption of data. T-HIBE can then be used to securely transport the symmetric session key from one IoT device to another.
2) We shall use a three-message pass between the two IoT devices, Device $\mathcal{A}$ from user-domain 1 and Device $\mathcal{B}$ from user-domain 2, to agree on a session-key and using T-HIBE as the key-transport protocol, as shown in Fig 5.
3) The first IoT device (Device $\mathcal{A}$) will generate a random key (α) and a random nonce (r_1) and encrypt this using T-HIBE and sends it to Device $\mathcal{B}$: $E_{\text{T-HIBE}_\mathcal{B}}(\alpha \,||\, r_1)$
4) Device $\mathcal{B}$ will respond back to the initial request by sending the received nonce (r_1) along with a randomly generated key (β) and another random nonce (r_2), encrypted using T-HIBE credentials of Device $\mathcal{A}$: $E_{\text{T-HIBE}_\mathcal{A}}(\beta \,||\, r_1 \,||\, r_2)$
5) After the first successful exchange, Device $\mathcal{A}$ will send back the nonce received in the last message (r_2) back to Device $\mathcal{B}$ in the clear, hence confirming mutual entity authentication.
6) Both the device will now use the shared-secret key $H(\alpha \,||\, \beta)$ to communicate between them and to secure the actual data communication. By doing this, both parties have key-control and mutual key-confirmation.
7) After the short session has concluded, the session-key can be destroyed and re-established again in the future if needed. In this way, we are using less computational overhead than when all transmissions were encrypted using only HIBE.

VI. Practical Discussion

A. Realisation of the root PKGs

In our scheme, the PKG(s) are responsible for initializing the setup and needs to run only once. Although we have evaluated the feasibility of running the PKGs on a microcontroller in Sect VII, it can also be realized on multiple cloud service providers. Our scheme does not require the PKG(s) to be always online and only be recalled during re-sharing of the master-key to a new PKG. One of the advantages a cloud PKG has from a local one, apart from faster execution times, is operating remotely and not being tied down to a specific location. Ideally, the n root PKGs will be hosted on different cloud service providers such as Amazon Web Services, Google Cloud and Microsoft Azure. It will only correlate with t other PKGs once to generate the parameters. Once this is done, the domain users will contact any active t out of the total n cloud PKGs to get their share of their private-key. The domain user will then collect all the received shares required and will locally compute the private-key. In turn, the users will become the domain PKGs of their respective domains and compute private-keys for the IoT devices present below. No interaction between the IoT devices and the root PKGs are required, and the system can be functional even if all the root PKGs goes offline.

In T-HIBE, it is also possible to add more PKGs later on if one or a few of the existing PKGs quits or change the number of shares required to compute the secret. With proper re-sharing, a (t, n) scheme can be converted to a (t', n') scheme with the final secret being the same, as long as the final total number of shares n' does not become greater than the field's modulo prime q. To re-share the existing master-secret and give the new PKG its shares, the other active PKGs needs to perform the following actions.

1) The new PKG(s) P_z makes itself known to t, or greater than t, other active PKGs.
2) Each PKG P_i, where $1 \leq i \leq t$ and $i \neq z$, that are a part of the original (t, n) scheme selects a random polynomial, similar to Equation (3), $f_i(y) \in \mathbb{Z}_q$ of degree $t' - 1$ of the form:

$$f_i(y) = a_{i0} + a_{i1}x + a_{i2}y^2 + ... + a_{i(t'-1)}y^{t'-1} \quad (9)$$

and set its old share s_i as a_{i0}, that is, $f_i(0) = a_{i0} = s_i$
3) Every PKG P_i will compute and send $A_{ik} = a_{ik}P$ for $k \in [0, t' - 1]$ to all other PKGs.
4) For each PKG P_z in the new scheme, each PKG P_i will compute the share $s_{iz} = f_i(z) \mod q$ and send it to PKG P_z via a secure channel, for all $i, z \in [1, t']$
5) Each PKG P_z can now verify the all the $t' - 1$ shares received from the other PKGs by computing

$$s_{iz}P = \sum_{k=0}^{t'} z^k A_{ik} \quad (10)$$

$$\text{for all } i, z \in [1, t']$$

6) If the verification succeeds, the new shares s'_z are generated by computing

$$s'_z = \sum_{i=0}^{t'} s_{iz} L_i(0) \mod q \quad (11)$$

Where $L_i(0)$ is defined as in Equation (5)

7) The originally computed final secret s still remains the same, while the scheme transforms from (t, n) to (t', n').
8) The re-sharing protocol can be run as many number of times as possible, as long as the total number of shares n is less than the field's prime modulo q, i.e., $n \leq q - 1$.

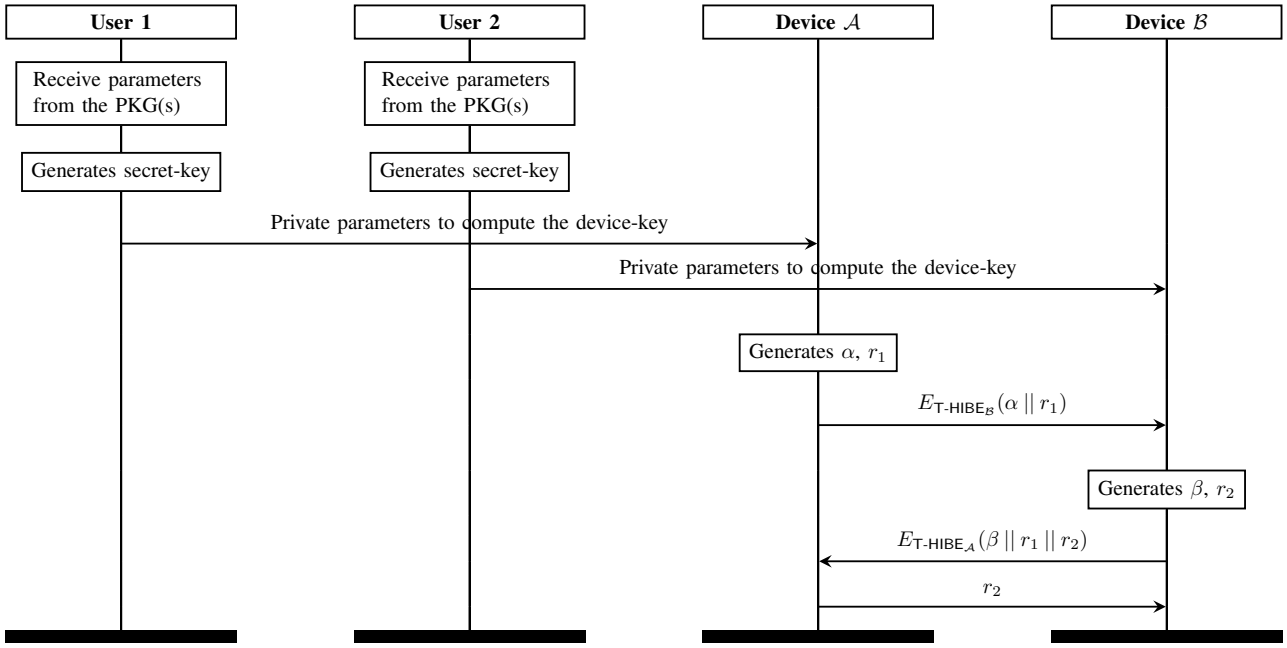

Fig. 5: Key-transport using T−HIBE

B. Temporary identifiers

One of the reasons IBE is interesting to use in an IoT setting is that it solves the need to verify the authenticity of a public key. However, similarly to PKI, one needs to consider revocation. IoT devices can get compromised, and one needs to avoid setting up a secure connection with a compromised device. This is not an easy problem to solve in IBE, as the public key is equal to the identity of a device, and one cannot revoke one's identity. However, it is possible to use temporary identities to overcome this problem. These temporary identities are the identity of the device along with a time period. Instead of the public-key being only the identity of the device, a short time-span can be added as a temporal identifier in the form of "ID || timestamp".

Once the time period has passed, this temporary identity has expired by default. This solves the aforementioned problem, as a temporary identifier (and hence the public key) can no longer be used after the time period has expired. After this period, the IoT device needs to obtain from the PKG on the layer above a fresh private-key. This private-key corresponds to the new public-key that will be used by the IoT device, which equals the new temporary identifier for the new time period. Unlike the PKI scheme, the other devices do not need to obtain new certificates and verify their authenticity every time the public-key changes. Hence the other devices do not need to communicate with the PKG(s) and can verify the public-key themselves.

The duration of this timestamp, that is, the validity of the public-key, is essentially a trade-off between security and efficiency. A long time period will require fewer key updates hence improving efficiency. However, it will also have a larger impact when a device gets compromised or the key gets leaked, affecting security. Increasing the frequency of key updates will increase the overall security. However, it also means, depending on which layer the key has been updated, it might need to re-run the entire protocol to generate new sets of key pairs for the updated keys. If this is done on a higher level, all the devices/users present below in the hierarchy also need to update their keys as the original key changes. This effectively decreases efficiency. Gradual key exposure is still a problem in most crytosystems [6]. Periodically refreshing the key or dividing the key into multiple sub-keys such that leakage of a few sub-keys will not leak any information on the original key are the only natural ways to tackle the problem. We argue that as we have already implemented a threshold structure at the root level, we have essentially solved the key-leakage problem at that level, and we do not need to deploy temporary identifiers at all layers/levels in the system and only where it is necessary. Also, the time windows on the timestamp should be different for different levels to balance efficiency and security. Frequent key refreshes in the user domain level will, in turn, mean the IoT devices also need to be refreshed with the new user's private-key. Hence, as the IoT devices are at the lowest level of the hierarchy and the most susceptible to leakage and compromise, devices on this layer should have the shortest time window compared to the other levels. The keys in the user level should also be periodically refreshed by the root PKGs but at a far higher time interval. The time windows and all the users' devices and IoT devices are a part of the public parameters, and this information is available to all devices beforehand.

TABLE I: Performance evaluation

T-HIBE phases	Execution time	Clock cycles
Setup generation (1,1)	30 ms	5,390,105
Key extraction (1,1)	230 ms	41,381,743
Setup generation (3,3)	410 ms	73,781,909
Key extraction (3,3)	640 ms	115,212,481
1-level encryption	560 ms	100,792,511
1-level decryption	249 ms	44,889,043
2-level encryption	680 ms	122,414,327
2-level decryption	590 ms	106,229,541
AES-CBC encryption	0.021 ms	3774
AES-CBC decryption	0.036 ms	3480

C. Delegation and Future Messages

In T-HIBE, it is also possible to delegate the task to be the manager device of the user for a limited period, for example, either when the user goes for a vacation or decides to rent his apartment for a brief while. The user can issue fresh keys with a revocable timestamp for the delegated user. The new delegated user can use these keys to establish new temporary keys for the devices below. The original user can also revoke the access at will.

Due to the inherent property of IBE, it is possible to send a message to a device intended only to be read at a future date [4]. As the public-key of a device is its identity and a set date, the sender can encrypt a message with a future date and send it. If that receiver device receives fresh keys from its user for that new date, it can then decrypt it as it will receive the corresponding decryption key for that specific public-key. If it receives the message earlier, as it does not have the future decryption key corresponding to that future public-key, it cannot decrypt that ciphertext.

VII. Implementation

In this section, we evaluate the feasibility of running our solution on IoT devices. We tested our setup on three ARM Cortex-M4 microcontrollers. Although we included the key setup and key extraction phase in our execution, these numbers can be ignored if needed, as the root PKG(s) will run these phases seldomly, and it can be assumed the root PKG to be a powerful high-end device or hosted on a cloud platform. We used the ARM Cortex-M4 processor on the DISCO-F469NI discovery board and implemented our setup using ARM Mbed-CLI v1.8.3 and GCC-ARM v6.3.1 as the compiler. Our testing module was built on the RELIC-toolkit [2], a modern cryptographic library for the embedded devices. We used the pairing friendly BN-254 curve to achieve a 128-bit security level. The microcontroller achieved a setup phase generation in 30 milliseconds and private key extraction in 230 milliseconds per user for a single PKG. A threshold (3,3) scheme took an average 410 milliseconds and 640 milliseconds for setup generation and key extraction, respectively. The encryption and decryption phases without hierarchy took 560 milliseconds and 250 milliseconds, respectively, as shown in Table I. A 2-level hierarchical setup, as envisioned in our paper, takes 680 ms for encryption and 590 ms for decryption. After this stage, the message will encapsulate a 128-bit symmetric key jointly generated and agreed upon by both the devices and used as a session-key between the devices. AES-128 encryption and decryption times for a Cortex-M4 microcontroller are around 21 microseconds and 36 microseconds, respectively. As with increasing depth in the hierarchy, the ID-length will increase, and so will the ciphertext length. Hence, increasing the hierarchical depth will have some impact on the execution time. Hence we demonstrate that, although with hierarchical depth efficiency decreases, it is not substantial for a 3-layered approach as stated in our work. Also, increasing the number of devices or users per hierarchical level will have no impact on performance. Although pairing-based cryptography is a resource-intensive operation, we find our implementation results modest, hence stating it can be run on an IoT device.

VIII. Related Work

Due to the heterogeneous nature of the IoT ecosystem, it is imperative for a key management solution to allow for secure communication, with strong confidentiality and integrity, between devices outside their trust domains. It should be possible for the manager of the trust domain to easily revoke keys or delegate access to other devices. There should not be a centralized hub that is always assumed to be trusted and act as a single point of failure, and the entire protocol should be lightweight and scalable enough to be run on low-powered IoT devices. Unfortunately, a lot of these requirements are missing in the current state-of-the-art. Several PKI-based solutions have been put forward. Gehrmann et al. [11] proposed an idea to deploy a *personal CA* in each security domain that is managed by a user. This personal CA could, for example, be the smartphone of the user or another trusted personal device. Within each domain, the personal CA can then issue certificates to any IoT devices in the network. To realize a multi-security domain scenario, one could rely on one root CA that authenticates each of the personal CA's underneath it. This way, each IoT device has a certificate that it can use to authenticate its permanent public key during a key establishment protocol with another IoT device. However, revocation is not always very intuitive, as the revocation lists need to be distributed among all the devices and checked before every key-establishment. Also, signing is a memory-intensive operation, and the devices always need to look up a public-key and verify its signature. [15] proposed an ECC-based key-establishment solution that involves a centralized entity known as the Registration Authority (RA), which is assumed to be honest and trusted. Several ABE-based key-management solutions have been proposed, including CP-ABE [10] and KP-ABE [21]. However, in both the works, the authors did not tackle to problem of key-escrow and assumed the root PKG to be honest and trusted at all times. Also, previous work has shown ABE being a cryptographically expensive

operation to execute, where encrypt and decrypt functions take a few seconds to compute on an IoT device [22]. These works are also not very scalable and do not work well with an increasing number of devices. [1] proposed a decentralized batch-based group key management protocol with full forward secrecy and fully scalable. However, due to no authenticity checks, this is susceptible to replay attacks, hence insecure. Moreover, it is not possible to revoke any distributed key. Previous works involving IBE in and IoT [7], [16] do not solve the inherent key-escrow problem and put full trust in the PKG. Additionally, these systems are neither scalable nor allow delegation. Also, it is expensive to use IBE for all the operations in a resource constraint device, such as IoT. Our work tries to solve these problems by only using T-HIBE as a key-transport mechanism, hence encrypting/decrypting only once. In Table II, we provide a comparison with the stated previous works.

TABLE II: Comparison with other relevant works

Protocol	Secure	Scalable	Efficient	Revokable	Solves key escrow /centralization
Personal CA [11]	●	●	❍	❍	●
ECC [15]	●	●	●	●	❍
CP-ABE [10]	●	●	❍	●	❍
KP-ABE [21]	●	●	❍	❍	❍
Group-key [1]	❍	●	●	❍	●
IBE [7], [16]	●	❍	●	●	❍
This work [T-HIBE]	●	●	●	●	●

IX. Conclusion

One of the security challenges in decentralized IoT ecosystems is the establishment of secure connections between IoT devices from multiple security domains. To tackle this problem, we proposed T-HIBE, a Threshold-based Hierarchical Identity-Based Encryption scheme for IoT. T-HIBE allows efficient key management in a multi-security domain setting. We tackled the key-escrow problem of HIBE by using multiple root PKGs instead of one and hence provide additional trust and transparency. We argue that having no certificate overhead with almost similar performance makes T-HIBE a robust and practical framework and an interesting alternative to conventional PKI solutions. Our solution is also scalable and can be realized on an embedded IoT device. We evaluated our scheme on the popular ARM Cortex-M4 microcontroller and showed that it is viable for IoT devices.

Acknowledgments

This work was supported in part by CyberSecurity Research Flanders with reference number VR20192203, by the Research Council KU Leuven C1 on Security and Privacy for Cyber-Physical Systems and the Internet of Things with contract number C16/15/058, the FWO SBO project SPITE, and by the European Union's Horizon 2020 Research and innovation program under grant agreement No. 826284 (ProTego).

References

[1] Abdmeziem, M.R., Tandjaoui, D., Romdhani, I.: A decentralized batch-based group key management protocol for mobile internet of things. In: 2015 IEEE International Conference on Computer and Information Technology. pp. 1109–1117 (2015)

[2] Aranha, D.F., Gouvêa, C.P.L., Markmann, T., Wahby, R.S., Liao, K.: RELIC is an Efficient LIbrary for Cryptography. https://github.com/relic-toolkit/relic

[3] Boneh, D., Boyen, X., Goh, E.J.: Hierarchical identity based encryption with constant size ciphertext. Cryptology ePrint Archive, Report 2005/015 (2005), https://eprint.iacr.org/2005/015

[4] Boneh, D., Franklin, M.: Identity-based encryption from the weil pairing. In: Kilian, J. (ed.) Advances in Cryptology — CRYPTO 2001. pp. 213–229. Springer Berlin Heidelberg, Berlin, Heidelberg (2001)

[5] Boyen, X., Waters, B.: Anonymous hierarchical identity-based encryption (without random oracles). Cryptology ePrint Archive, Report 2006/085 (2006), https://eprint.iacr.org/2006/085

[6] Canetti, R., Dodis, Y., Halevi, S., Kushilevitz, E., Sahai, A.: Exposure-resilient functions and all-or-nothing transforms. In: Advances in Cryptology - EUROCRYPT 2000. Lecture Notes in Computer Science, vol. 1807, pp. 453–469. Springer (2000)

[7] Chen, W.: An ibe-based security scheme on internet of things. In: 2012 IEEE 2nd International Conference on Cloud Computing and Intelligence Systems. vol. 03, pp. 1046–1049 (2012)

[8] Cocks, C.: An identity based encryption scheme based on quadratic residues. In: Proceedings of the 8th IMA International Conference on Cryptography and Coding. p. 360–363. Springer-Verlag, Berlin, Heidelberg (2001)

[9] Dodis, Y., Yung, M.: Exposure-resilience for free: the hierarchical id-based encryption case. In: First International IEEE Security in Storage Workshop, 2002. Proceedings. pp. 45–52 (2002)

[10] Fischer, M., Scheerhorn, A., Tönjes, R.: Using attribute-based encryption on iot devices with instant key revocation. In: 2019 IEEE International Conference on Pervasive Computing and Communications Workshops (PerCom Workshops). pp. 126–131 (2019)

[11] Gehrmann, C., Nyberg, K., Mitchell, C.J.: The personal ca - pki for a personal area network. In: Mobile and Wireless Communications Summit 2002. pp. 31–35. IST (2002)

[12] Gentry, C., Silverberg, A.: Hierarchical id-based cryptography. Cryptology ePrint Archive, Report 2002/056 (2002)

[13] Horwitz, J., Lynn, B.: Toward hierarchical identity-based encryption. In: Knudsen, L.R. (ed.) Advances in Cryptology — EUROCRYPT 2002. pp. 466–481. Springer Berlin Heidelberg, Berlin, Heidelberg (2002)

[14] Joux, A., Nguyen, K.: Separating decision diffie-hellman from diffie-hellman in cryptographic groups. Cryptology ePrint Archive, Report 2001/003 (2001), https://eprint.iacr.org/2001/003

[15] Liu, J., Xiao, Y., Chen, C.P.: Authentication and access control in the internet of things. In: 2012 32nd International Conference on Distributed Computing Systems Workshops. pp. 588–592 (2012).

[16] Markmann, T., Schmidt, T.C., Wählisch, M.: Federated end-to-end authentication for the constrained internet of things using ibc and ecc. SIGCOMM Comput. Commun. Rev. **45**(4), 603–604 (Aug 2015)

[17] Menezes, A.J., Okamoto, T., Vanstone, S.A.: Reducing elliptic curve logarithms to logarithms in a finite field. IEEE Transactions on Information Theory **39**(5), 1639–1646 (1993)

[18] Pedersen, T.: A Threshold Cryptosystem without a Trusted Party. In: Advances in Cryptology — EUROCRYPT '91. pp. 522–526.

[19] Shamir, A.: How to share a secret. Commun. ACM **22**(11), 612–613 (Nov 1979). https://doi.org/10.1145/359168.359176

[20] Shamir, A.: Identity-based cryptosystems and signature schemes. In: Blakley, G.R., Chaum, D. (eds.) Advances in Cryptology. pp. 47–53. Springer Berlin Heidelberg, Berlin, Heidelberg (1985)

[21] Touati, L., Challal, Y.: Collaborative kp-abe for cloud-based internet of things applications. In: 2016 IEEE International Conference on Communications (ICC). pp. 1–7 (2016)

[22] Wang, X., Zhang, J., Schooler, E.M., Ion, M.: Performance evaluation of attribute-based encryption: Toward data privacy in the iot. In: 2014 IEEE International Conference on Communications (ICC). pp. 725–730.