Smart Home Security: A Distributed Identity-based Security Protocol for Authentication and Key Exchange

M. Mazhar Rathore, Elmahdi Bentafat, and Spiridon Bakiras
Division of Information and Computing Technology
College of Science and Engineering, Hamad Bin Khalifa University
Email: {mrathore, ebentafat, sbakiras}@hbku.edu.qa

***Abstract*—Smart home technology is gaining popularity among end-users, as it allows them to remotely control a variety of devices in their homes. Such systems have obvious benefits in terms of automation, but at the same time pose significant threats to home owners if the underlying communications are not secure. To this end, we introduce a novel security protocol that simplifies the pairwise authentication and key exchange among smart home devices. The protocol leverages identity-based cryptography (IBC), thus relaxing the requirement for storing and managing public key certificates. Furthermore, to mitigate the risks associated with centralized key generation in IBC, we opt to generate the private keys in a distributed manner, involving all smart devices. We implemented our protocol on Raspberry Pi 3 devices and demonstrate its efficiency in terms of both computational and communication cost.**

I. Introduction

The Internet of Things (IoT) era has had a major impact on home automation. As a result, today's *smart homes* are equipped with numerous IoT devices, including security cameras, smart locks, smart thermostats, etc. Some devices are sensors that measure a variety of physical phenomena, while others are controlled actuators that react to changing environments. All these devices are interconnected via a home area network (HAN) and interact with the owner through a smartphone application. For example, the owner may receive an alert when there is motion in front of a security camera or remotely set the desired room temperature on the thermostat. Nevertheless, the benefits of smart home systems may be outweighed by the potential risks to home owners, if the underlying communications are not secure. Indeed, without proper authentication and end-to-end encryption protocols, an intruder could easily eavesdrop on the exchanged messages and analyze the owner's daily routine or, even worse, replay messages to manipulate certain IoT devices (e.g., unlock the front door).

To this end, transport layer security (TLS) is the de facto protocol for secure communications that is based on the existing public key infrastructure (PKI). In a smart home environment, TLS would necessitate that every IoT device stores the public key certificate of every other device in the HAN. These certificates are issued and signed by a trusted certification authority (CA) and serve as a proof-of-identity for the underlying devices. However, PKI has certain limitations in the context of IoT devices. First, IoT devices have limited storage and computational capabilities, so storing and verifying the certificates may incur a considerable overhead. Second, with potentially tens of billions of IoT devices on the market, the communication and computational cost at the CA level will increase significantly [1]. Finally, the vast number of IoT device certificates will complicate the certificate revocation process.

A promising alternative to PKI is identity-based cryptography (IBC) [2]. Under IBC, there is no certificate associated with a networked device. Instead, the device's unique ID, such as its name or MAC address, serves as the public key that can be used by other devices to communicate with it in a secure manner. More specifically, IBC requires a trusted key generation center (KGC) that generates and distributes a common public key to all devices. In addition, the KGC generates a unique private key for each device in the network, which is computed with the device's unique ID as an input. When Alice wants to send a secure message to Bob, she uses Bob's ID to encrypt the message, which can only be decrypted by Bob's private key. In a smart home environment, the KGC would typically be the home owner's device (e.g., smartphone) that would generate a private key for any new IoT device that is installed. Furthermore, to avoid expensive IBC operations for every transmitted message, the devices could leverage IBC to securely establish short-term session keys through a Diffie-Hellman key exchange protocol. Several IBC-based protocols for IoT networks have been proposed in the literature, including Ref. [3], [4], [5], [6], [7].

Nevertheless, the main drawback of IBC is its reliance on a trusted KGC. If compromised, the KGC would disclose all the devices' private keys, thus giving the adversary full control over the smart home appliances. This places an enormous burden on the security of the KGC device. On one hand, we cannot rely on the average user to enforce stringent security controls on their smartphones, as they may lack the technical knowledge to do so. On the other hand, placing the KGC at the smart home service provider's servers (e.g., Nest) would create a single point of failure and a clear target for cyber attacks. To this end, we propose a smart home security solution based on a distributed implementation of Boneh and Franklin's [8] protocol that utilizes pairing-based cryptography. In particular, our method removes the KGC requirement, by computing the private keys in a distributed manner with the participation of all IoT devices. As a result, even if one or more devices are compromised, the adversary can not derive any information regarding the private keys of the uncorrupted devices.

Furthermore, we propose a novel method for authenticated key exchange that leverages the properties of pairing-based cryptography to (i) authenticate any pair of devices on the network, and (ii) establish a Diffie-Hellman session key. The key exchange protocol is very efficient, requiring just two multiplications and one broadcast message per device. After that, a session key between two nodes is computed with a single pairing computation. We implemented our protocols on Raspberry Pi 3 devices and illustrate their scalability and practicality in a smart home environment. To summarize, the main contributions of our work are as follows:

- We introduce a distributed identity-based security protocol for smart homes that mitigates the risks associated with a trusted KGC. The protocol maintains the security of the honest devices' private keys, even when the admin device is compromised.
- We propose a novel authenticated key exchange protocol that greatly simplifies the establishment of short-term session keys between any pair of smart home devices. The simplicity and efficiency of the protocol facilitate frequent key updates for enhanced security.
- We implemented all protocols on Raspberry Pi 3 devices and test their performance with a large number of devices on a WiFi-based HAN.

The rest of the paper is organized as follows. Section II presents the related work and Section III describes the basic cryptographic primitives utilized in our methods, namely bilinear maps and identity-based cryptography. Section IV introduces our smart home security protocol and Section V analyzes its security. Section VI discusses our implementation details and also presents the results of our experimental evaluation. Finally, Section VII concludes our work.

II. Related Work

Shamir was the first to introduce the notion of identity-based cryptography in 1984 [2]. The application scenario was an email system that doesn't rely on public key certificates. However, the first practical construction of identity-based encryption (IBE) was due to Boneh and Franklin in 2001 [9]. Several IBE constructions have been reported in the literature, with most of them employing pairing-based cryptography over elliptic curves [8], [10], [11]. On the other hand, Cocks' IBE protocol [12] is based on quadratic residues. Advances in lattice-based cryptography have also led to the first lattice-based IBE scheme by Ducas et al. [13] in 2014. This protocol was later implemented by McCarthy et al. [14] as a fully functional C library. Nevertheless, lattice-based cryptography is very costly in terms of communication and storage requirements and is, thus, not practical for IoT devices.

Instead, the application of identity-based protocols in IoT and sensor network environments have mostly considered compact elliptic curve constructions. For example, IBAKA [3] leverages the Boneh and Frankin scheme [8] to perform Diffie-Hellman key exchanges [15] in IoT networks. Their method necessitates two pairing operations and three point-scalar multiplications, which are both costly in terms of computational cost. Tiny IBE [4] is a lightweight authenticated key distribution protocol for heterogeneous sensor networks that (i) avoids expensive pairing computations and (ii) requires just two messages to establish a session key between two nodes. Similarly, Yao et al. [16] also avoid the use of pairing operations, by leveraging an attribute-based encryption scheme for IoT environments. Finally, Mao et al. [5] introduce an IBE-based protocol for the secure communication of IoT nodes that employs fuzzy logic.

In the context of smart homes, Nicanfar et al. [6] propose an IBC-based scheme for key management. In particular, they introduce an efficient private key refreshment method, while also providing multicast keys that are sometimes necessary for HANs. In a subsequent work, the same authors improve their solution [17] by reducing the number of exchanged packets and the number of protocol steps. Jacobsen et al. [7] focus on optimizing the bootstrapping phase of wireless devices in HANs, by leveraging IBC to establish session keys. In a different study [18], Gao introduces new techniques for incorporating biometrics in the authentication phase of smart grids. The author proposes the use of fingerprints to improve the users' privacy in smart grid communications.

Our work is also related to threshold cryptosystems that are based on Shamir's secret sharing scheme [19]. Threshold cryptosystems distribute private keys among several parties, and require multiple devices in the network to participate in the encryption or signing of messages. For instance, Gennaro et al. [20] propose a protocol where the key is divided into n secrets, and key reconstruction necessitates the combination of any $t+1$ out of n secrets. On the other hand, producing a digital signature requires the cooperation of $2t+1$ parties, but there is no need to reconstruct the key. A more efficient scheme is due to Gennaro et al. [21] that targets digital signatures in bitcoin wallets. Their construction is based on the elliptic curve digital signature algorithm, and does not require an honest majority of devices. Threshold cryptosystems have also been proposed for the RSA protocol [22]. Nevertheless, threshold cryptosystems are expensive and, thus, not suitable for resource constrained IoT environments.

III. Preliminaries

Identity-based cryptography is based on bilinear maps on prime order groups over elliptic curves. Specifically, given two groups $\mathbb{G}$, $\mathbb{G}_T$ of the same prime order q, a bilinear map $e : \mathbb{G} \times \mathbb{G} \rightarrow \mathbb{G}_T$ satisfies the following properties[1]:

1) It is *computable*, i.e., given $U, V \in \mathbb{G}$, there is a polynomial time algorithm for computing $e(U, V) \in \mathbb{G}_T$.
2) It is *bilinear*, i.e., for any $U, V \in \mathbb{G}$ and $a, b \in \mathbb{Z}_q$, $e(aU, bV) = e(U, V)^{ab}$.

[1]Throughout the paper, we use uppercase characters to represent elliptic curve points, and lowercase characters to represent scalars.

3) It is *non-degenerate*, i.e., if P is a generator of $\mathbb{G}$ then $e(P, P)$ is a generator of $\mathbb{G}_T$.

Under the Boneh-Franklin cryptosystem [8], the KGC first selects a random private master key $K_s = s \in \mathbb{Z}_q^*$, and sets the public key $K_p = sP$, where P is a generator of group $\mathbb{G}$. Then, every user in the network is assigned a public key, which is a string representing the user's ID (e.g., email address). Given an $ID \in \{0, 1\}^*$, the KGC computes $Q_{ID} = H(ID)$ and assigns a private key $D_{ID} = sQ_{ID}$ to that particular user. $H(\cdot)$ is a hash function that maps an arbitrary string to a point of the group $\mathbb{G}$ on the elliptic curve. The Boneh-Franklin scheme further defines functions for public key encryption and decryption, where (K_p, Q_{ID}) and D_{ID} are used for encryption and decryption, respectively.

Nevertheless, in a smart home environment, identity-based encryption is not sufficient, because both parties have to authenticate each other during message exchange. As such, a message from device i to device j would necessitate (i) an identity-based encryption with j's public key and (ii) an identity-based signature [23] with i's private key. This would incur a significant computational cost at both parties and is, thus, impractical for smart homes. Instead, in this work, we leverage the devices' private keys to compute pairwise shared secrets that are used to perform authenticated Diffie-Hellman key exchanges of short-term session keys. In particular, for devices i, j with public/private key pairs (Q_i, D_i) and (Q_j, D_j), respectively, the shared secret is

$$e(D_i, Q_j) = e(Q_i, D_j) = e(Q_i, Q_j)^s$$

Note that $e(Q_i, Q_j)^s$ can only be computed by devices i and j that own the corresponding private keys.

Finally, since our protocols target smart home devices with limited computational capabilities, it is worth mentioning the complexity of the various elliptic curve operations involved. As we will show in Section VI, the most time consuming function is map-to-point (H), followed by the pairing operation e and the point-scalar multiplication. Our proposed protocols aim to minimize the number of these operations.

IV. Smart Home Security

This section presents the details of our smart home security solution. We begin by describing the overall system architecture, followed by a series of protocols for new device registration, distributed key generation, and authenticated key exchange.

A. Smart Home Architecture

A smart home consists of a series of sensors and actuators, connected through a communications infrastructure (Fig. 1). Sensors sense the environment, whereas actuators react based on the current conditions. For example, temperature sensors measure the temperature at different locations inside the home, while the AC may adjust itself according to the current temperature. The home owner interacts with all devices through an *admin* device, which is typically an app running on the owner's smartphone. To simplify the deployment and operation of the HAN, we assume that all communications are done through a wireless technology, such as Bluetooth or WiFi.

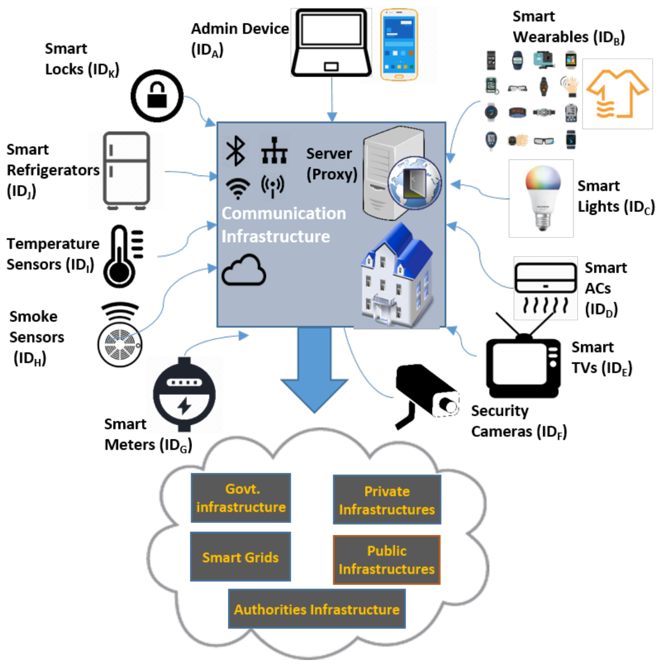


Fig. 1. Smart home architecture

With few exceptions, most smart devices do not need to exchange messages with a device other than the admin. On the other hand, the admin device should always be able to send/receive messages (such as alerts or commands), even when it is not located inside the smart home. Therefore, we assume that an external server with a known IP (i.e., the smart home service provider) acts as a proxy to transfer messages between the devices when they are outside the HAN. In particular, all home devices establish TLS connections to the remote server, which then receives and forwards each message to the corresponding destination. Nest's Smart Home Hub[2], developed by Nest Lab and now purchased by Google, also works in a similar fashion.

The goal of this work is to secure the communications among all smart devices. Specifically, we want to guarantee (i) message confidentiality and (ii) pairwise authentication for the sender and recipient of each message. Both properties are vital for the privacy and security (cyber and physical) of the home owner. For instance, unencrypted messages may reveal the owner's daily routine (at least to the smart home service provider), while insufficient authentication mechanisms may allow an adversary to open the front door by simply replaying a previously recorded message.

B. Threat Model

We consider two types of adversaries in our work:

- **Eavesdropper**: This type of adversary has access to all exchanged messages, and may perform a ciphertext-only

[2]https://nest.com/

attack to retrieve the underlying decryption keys. For example, the adversary may be an intruder that gained unauthorized access to the WiFi network, or the smart home service provider that forwards messages on behalf of the smart devices.

- **Malicious**: A malicious adversary has access to all communications, but may also launch a number of active attacks. For instance, the adversary may try to impersonate an IoT device, divert from the specified protocols, replay old messages, or attempt to launch man-in-the-middle (MITM) and denial-of-service (DoS) attacks.

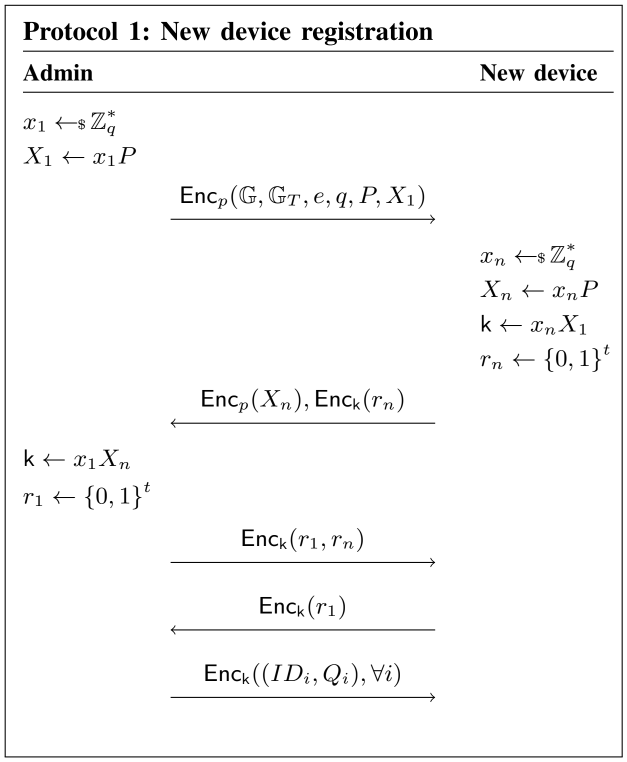
We consider all IoT devices as eavesdroppers, while any device that is compromised by the adversary is assumed to be malicious. Finally, we assume that all adversaries run in polynomial time and are, thus, not able to break the cryptographic protocols.

C. New Device Registration

We assume that every device i in the HAN has a unique identifier ID_i (e.g., *SmartTV-LivingRoom*) that is assigned to it by the home owner through the admin device. As explained in Section III, $Q_i = H(ID_i)$ is a point on an elliptic curve that serves as the public key of that device. We also assume that there is a way for the new device to select a temporary password p that is shared with the admin device. As an example, the device could have a small screen to generate and display the password on-demand, or the password could be hardcoded by the manufacturer and written on the device's documentation.

Fig. 2 illustrates the detailed protocol for new device registration. We assume that there are currently $n-1$ registered IoT devices, where ID_1 is the admin's ID and ID_n is the ID assigned to the new device. The protocol is a straightforward implementation of the encrypted key exchange (EKE) protocol by Bellovin and Merritt [24], using Diffie-Hellman. Initially, the admin device uses the shared password p to encrypt (i) the public parameters of the elliptic curve groups and (ii) its share of the Diffie-Hellman key exchange, X_1. Next, the new device generates its own share of the Diffie-Hellman exchange, X_n, and computes the session key $\mathsf{k} = x_n X_1 = x_n x_1 P$. Finally, it encrypts X_n with p and also encrypts a random t-bit number r_n with k, and sends both values to the admin device. The admin device then computes the session key k, generates a random t-bit number r_1, and sends $\mathsf{Enc}_{\mathsf{k}}(r_1, r_n)$ back to the new device. Once device n replies with $\mathsf{Enc}_{\mathsf{k}}(r_1)$, the two devices are certain about the correctness of the session key. At that point, the admin device sends the encrypted list of all IDs/public keys (including the newly assigned ones for device n) that are currently registered in the smart home's network.

The registration protocol is clearly very efficient, and requires just two point-scalar multiplications from the IoT device. (We are not concerned with the computational cost at the admin device, because today's smartphones are much more powerful compared to the cheaper IoT devices.) Also, the admin device will compute Q_n on behalf of the new device, in order to relieve it from performing the costly map-to-point



Protocol 1: New device registration
Admin
New device
$x_1 \leftarrow_{\$} \mathbb{Z}_q^*$
$X_1 \leftarrow x_1 P$
$\mathsf{Enc}_p(\mathbb{G}, \mathbb{G}_T, e, q, P, X_1)$
$x_n \leftarrow_{\$} \mathbb{Z}_q^*$
$X_n \leftarrow x_n P$
$\mathsf{k} \leftarrow x_n X_1$
$r_n \leftarrow \{0,1\}^t$
$\mathsf{Enc}_p(X_n), \mathsf{Enc}_{\mathsf{k}}(r_n)$
$\mathsf{k} \leftarrow x_1 X_n$
$r_1 \leftarrow \{0,1\}^t$
$\mathsf{Enc}_{\mathsf{k}}(r_1, r_n)$
$\mathsf{Enc}_{\mathsf{k}}(r_1)$
$\mathsf{Enc}_{\mathsf{k}}((ID_i, Q_i), \forall i)$

Fig. 2. New device registration

operation. Nevertheless, the new device may opt to re-compute all public keys locally, in order to verify their correctness and/or identify a potentially malicious admin device.

D. Distributed Key Generation

At the heart of our smart home security solution is a protocol for computing the devices' identity-based private keys in a distributed manner, in order to relax the requirement for a centralized and trusted KGC. Recall that, under IBC, the private key D_i for device i is equal to sQ_i, where s is the private master key of the KGC. In our approach, we employ a secret sharing scheme, where each device i chooses a secret share s_i uniformly at random from $\mathbb{Z}_q^*$. The master key is then set implicitly (but never computed) as

$$s = \sum_{\forall i} s_i \bmod q$$

Next, to compute D_j, it suffices to collect its shares $s_i Q_j, \forall i$ from all devices and aggregate them as follows:

$$D_j = \sum_{\forall i} s_i Q_j = sQ_j$$

This simple protocol is depicted in Fig. 3, where $2 \leq k < n$, i.e., k represents all devices other than the admin and the new device n. The protocol is invoked right after a device registration operation, in order for the new device to get its IBC secret key and for the old devices to update their keys as well. We assume that every device k (except for n at this point) shares a short-term session key k_{1k} with the admin device, through the authenticated key exchange mechanism described

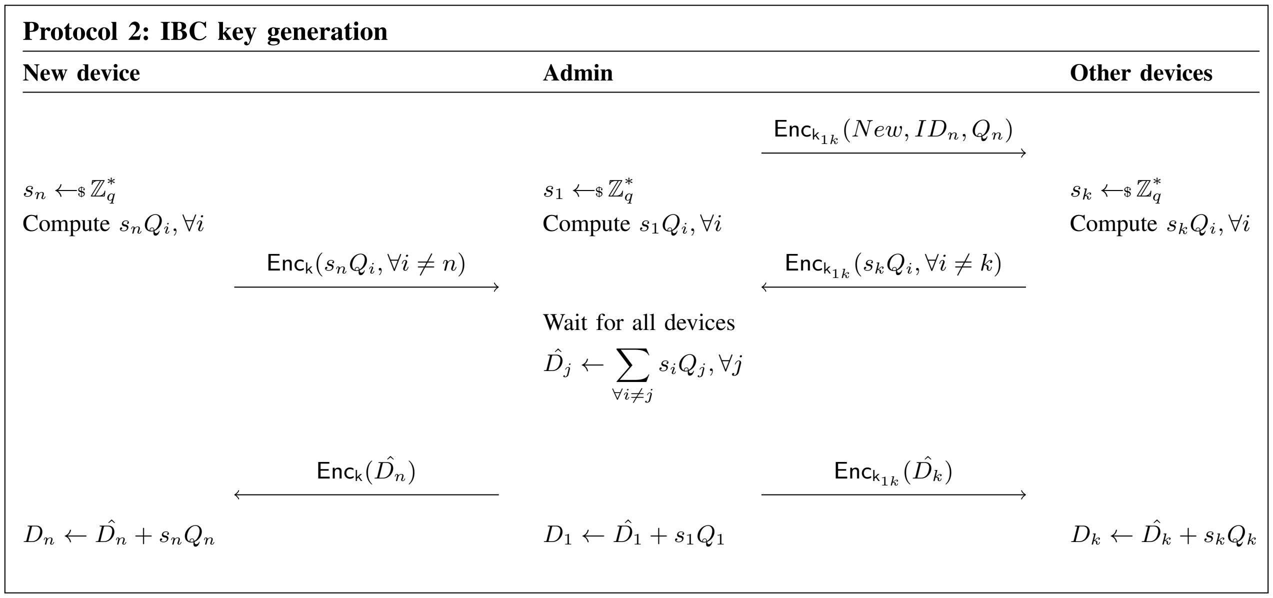

Protocol 2: IBC key generation
New device
Admin
Other devices
$\mathsf{Enc}_{\mathsf{k}_{1k}}(New, ID_n, Q_n)$
$s_n \leftarrow_\$ \mathbb{Z}_q^*$
Compute $s_nQ_i, \forall i$
$s_1 \leftarrow_\$ \mathbb{Z}_q^*$
Compute $s_1Q_i, \forall i$
$s_k \leftarrow_\$ \mathbb{Z}_q^*$
Compute $s_kQ_i, \forall i$
$\mathsf{Enc}_{\mathsf{k}}(s_nQ_i, \forall i \neq n)$
$\mathsf{Enc}_{\mathsf{k}_{1k}}(s_kQ_i, \forall i \neq k)$
Wait for all devices
$\hat{D}_j \leftarrow \sum_{\forall i \neq j} s_iQ_j, \forall j$
$\mathsf{Enc}_{\mathsf{k}}(\hat{D}_n)$
$\mathsf{Enc}_{\mathsf{k}_{1k}}(\hat{D}_k)$
$D_n \leftarrow \hat{D}_n + s_nQ_n$
$D_1 \leftarrow \hat{D}_1 + s_1Q_1$
$D_k \leftarrow \hat{D}_k + s_kQ_k$

Fig. 3. IBC key generation

in the following section. All communications involved in this protocol are encrypted with the most recent session keys. Note that, unlike other devices, n interacts with the admin device through the temporary key k constructed in the previous phase.

The first step is for the admin device to individually inform every existing device that a new round of distributed key generation is taking place. In addition, it shares with them the ID and public key of the new device n. After that, every device j computes the private key shares s_jQ_i for all devices $i \neq j$, and sends them to the admin device. It also computes its own private key share s_jQ_j, which is kept secret from other devices. Once the admin device gathers all the necessary information, it constructs the *partial* private keys $\hat{D}_j = \sum_{\forall i \neq j} s_iQ_j, \forall j$ and sends them to the corresponding devices. Finally, each device j adds its own secret share s_jQ_j to the partial key and computes the value of its IBC private key D_j.

The security of this algorithm stems from the fact that the private share of each device's IBC key is never revealed to any party. As such, even with the knowledge of partial key $\hat{D}_i$, it is impossible for the admin device to derive any information regarding D_i, or otherwise manipulate its value, if s_i is properly chosen from a uniform distribution. Furthermore, any attempt on behalf of the admin device to modify the values of the partial keys would result in failed key exchanges between the IoT devices (next section), because each device would have a different view of the master key s.

The computational cost of the protocol is linear to the number of IoT devices n, because it necessitates just n point-scalar multiplications per device (point additions have negligible cost). As such, it enables the home owner to invoke the protocol relatively frequently (e.g., once a week), in order to deter cryptanalytic attacks. Furthermore, the protocol can run for multiple new devices, assuming that they have all established a Diffie-Hellman key with the admin device beforehand. Finally, we should note that, the reason for aggregating the partial keys at the admin device is to avoid the exchange of pairwise session keys among devices that do not need to communicate with each other during normal operations. In a real-life scenario, only the admin device would typically communicate with all smart devices for the purpose of receiving alerts or interacting with the device.

E. Authenticated Key Exchange

Following the distributed key generation phase, the IoT devices must establish short-term session keys with (i) the admin device and (ii) any other smart device that they need to communicate with, in accordance with the underlying smart home configuration. The detailed protocol for two devices i and j is shown in Fig. 4. Initially, every device i chooses a secret key x_i uniformly at random from $\mathbb{Z}_q^*$, and broadcasts (in plaintext) the corresponding public key $X_i = x_iQ_i$ to all devices. After that, the session key k_{ij} can be computed by each device as

$$\mathsf{k}_{ij} = e(x_iD_i, X_j) = e(X_i, x_jD_j) = e(Q_i, Q_j)^{sx_ix_j}$$

The last three message exchanges verify that the two devices have a consistent view of the new session key.

Protocol 3 is secure, because only devices i and j are capable of computing the correct value of the session key. Indeed, k_{ij} is the product of a pairing operation, where one of the inputs is the private IBC key of the corresponding device. Therefore, even with public knowledge of X_i and X_j, an adversary has no advantage in guessing the key value. Essentially, Protocol 3 is a Diffie-Hellman key exchange protocol, where the base element is $e(Q_i, Q_j)^s$, i.e., the shared secret between i and j. As a result, a successful completion of the protocol implies that devices i and j are authenticated.

Protocol 3: Authenticated key exchange

$$
\begin{array}{lcl}
\textbf{Device } i & & \textbf{Device } j \\
\hline
x_i \leftarrow_{\$} \mathbb{Z}_q^* & & x_j \leftarrow_{\$} \mathbb{Z}_q^* \\
X_i \leftarrow x_i Q_i & & X_j \leftarrow x_j Q_j \\
& \xrightarrow{\text{Bcast: } X_i} & \\
& \xleftarrow{\text{Bcast: } X_j} & \\
\mathsf{k}_{ij} \leftarrow e(x_i D_i, X_j) & & \mathsf{k}_{ij} \leftarrow e(X_i, x_j D_j) \\
r_i \leftarrow \{0,1\}^t & & r_j \leftarrow \{0,1\}^t \\
& \xrightarrow{\mathsf{Enc}_{\mathsf{k}_{ij}}(r_i)} & \\
& \xleftarrow{\mathsf{Enc}_{\mathsf{k}_{ij}}(r_i, r_j)} & \\
& \xrightarrow{\mathsf{Enc}_{\mathsf{k}_{ij}}(r_j)} &
\end{array}
$$

Fig. 4. Authenticated key exchange

A nice feature of the protocol is its computational efficiency that facilitates very frequent key updates. Specifically, every device must perform two point-scalar multiplications, regardless of the total number of devices n, and then one pairing operation per key exchange. As mentioned previously, though, we do not expect an IoT device to share session keys with more than a handful of other devices.

V. Security Analysis

In this section we analyze the security of our proposed protocols, for two different types of adversaries.

A. Eavesdropper

An eavesdropper with access to all communication transcripts has no advantage in gaining any information regarding the underlying messages. This is due to the secure symmetric encryption protocols that are employed at all times. As evident in Section IV, the only plaintext messages appear in the authenticated key exchange protocol, where the devices broadcast their public keys. Nevertheless, this information does not reveal anything to the adversary, because he can not compute the corresponding private keys. The weakest key utilized in our protocols is the temporary key p that appears in Protocol 1 (new device registration). The reason is that the key should only consist of printable characters in order for the home owner to type it easily into the admin device. As such, p would not be as strong as a random 256-bit key, but this is not a major concern. For security, we simply want p to remain secret for a few milliseconds, just enough for the Diffie-Hellman key exchange to complete. Even if p is compromised at a later time, the only information that an adversary can retrieve is the description of the elliptic curve groups, which is usually public knowledge.

B. Malicious Adversary

Under a malicious adversary, a number of different active attacks are possible. We discuss the most standard ones below:

- **Man-in-the-middle**: MITM attacks take place during key exchange protocols, where the adversary tries to inject itself into the communication channel between two nodes, in order to gain the ability to decrypt and/or modify all messages. For a successful MITM attack, an adversary would have to defeat the underlying authentication mechanisms which, in our protocols, are the shared secrets p (Protocol 1) and $e(Q_i, Q_j)^s$ (Protocol 3). As such, our protocols are secure against MITM attacks, because it is computationally hard for an adversary to guess these values.
- **Replay**: Replay attacks are very common and their purpose is to capture and replay old messages, in order to force a device to repeat some actions (e.g., open the front door). Our protocols defend against such attacks by frequently changing the pairwise session keys. Furthermore, a straightforward defense against replay attacks that may be easily incorporated in our methods, is the use of *timestamps* in all encrypted messages.
- **Node impersonation**: To successfully impersonate a device, an adversary would have to compromise its private IBC key. Therefore, unless a device is physically compromised (i.e., all its secret keys are disclosed), such attacks are not feasible. A compromised admin device is very dangerous, because it can communicate with all other devices. However, even in this case, our distributed key generation algorithm protects the secrecy of all the remaining private keys. Finally, an additional security layer to protect against a potential admin impersonator, is to employ two-factor authentication techniques for critical applications.
- **Sybil**: A compromised admin device (impersonator) can launch a Sybil attack by registering fake devices in the smart home network. However, this is not a weakness of our protocols, since the admin device has the permission to install new devices. Enforcing two-factor authentication during new device registration can mitigate such attacks.
- **Denial-of-service**: DoS attacks are very hard to defend against, so our protocols simply include the mechanisms to detect them. In our case, a DoS attacker would try to interfere with the protocol execution in order to prevent the establishment of the session keys or the registration of new devices. For example, the adversary may send fake key shares to other nodes (through an impersonation attack) that lead to failed key exchange protocols, thus disrupting the network operation. Nevertheless, all our key exchange protocols employ key verification methods (by exchanging encrypted random values), so this type of attacks can be identified and reported to the home owner.

VI. Implementation and Evaluation

To test the feasibility of our solution in a smart home environment, we implemented all protocols on 10 Raspberry Pi 3 model B devices, with a 1.2 GHz CPU capable of running one thread per core. To emulate the more powerful admin device, we utilized a Linux desktop machine with a 3.0 GHz CPU capable of running two threads per core. The devices were connected over a WiFi network that carried other background traffic as well. All implementations were written in C, using Ben Lynn's PBC library[3] for the elliptic curve operations, the GMP library[4] for arbitrary precision arithmetic operations, and the OpenSSL library[5] for symmetric encryption. For security, we chose elliptic curve groups of order q, where q is a 256-bit prime. The symmetric cipher of choice was AES in GCM mode, using 256-bit keys.

Before executing the full protocols, we measured the CPU cost of the basic cryptographic operations on the two types of devices. The results are summarized in Table I. The first observation is that the Raspberry Pi 3 devices are significantly slower, up to one order of magnitude in some cases. Nevertheless, none of the operations takes more than 129 ms to complete. For real IoT devices, we expect these costs to be slightly larger, but still within reasonable values. The second observation regards the cost of the elliptic curve operations. Point additions are extremely fast on both machines, while the map-to-point function is by far the most expensive one. Recall, however, that the map-to-point operations (for generating the IBC public keys) are performed exclusively at the admin device. The majority of the elliptic curve operations invoked by the IoT devices are point-scalar multiplications, which require 40 ms of computation time. On the other hand, pairing operations are rarer, but they are significantly slower (82 ms). Finally, symmetric encryption, which is the most frequently used operation, is very efficient and takes just 0.012 ms on the Raspberry Pi devices.

TABLE I
CPU time (ms) for basic operations

Operations	Desktop	Raspberry Pi 3
Point addition	0.001	0.002
Point-scalar multiplication	6.041	39.694
Pairing	8.595	81.625
Map-to-point	18.568	128.194
Symmetric encryption (256 B)	0.001	0.013
Symmetric decryption (256 B)	0.002	0.012

Next, we invoked all three protocols and measured their execution times, while varying the number of participating devices n. We started with Protocol 1 that enables the installation of new IoT devices by the home owner, and the results are depicted in Fig. 5. For a single IoT device (solid line), the time is constant (around 150 ms) and independent of n, since it only interacts with the admin device. On the other hand, the response time at the admin device is linear in n, as indicated by the dotted line. When registering 10 devices in a row, the running time of the protocol is less than 1.5 sec.

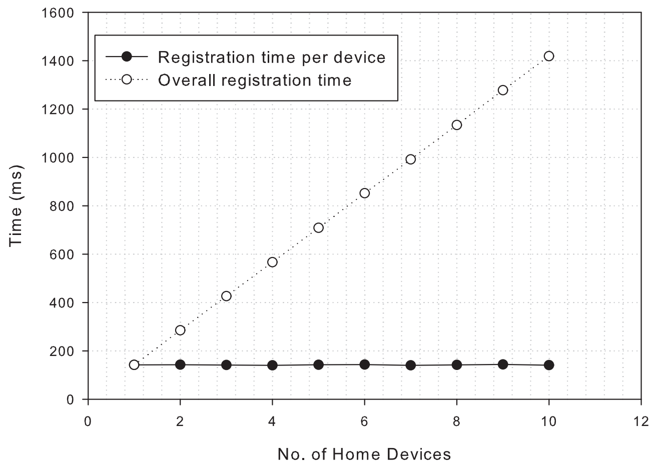


Fig. 5. Response time for new device registration

Fig. 6 illustrates the response time of our main protocol that generates the private IBC keys in a distributed manner. The cost grows slightly slower than linear, due to the offloading of computations from the slower IoT devices to the admin device (as explained in Section IV-D). In addition, the desktop machine can perform some parallel computations through threading. For one device, the response time is about 120 ms, while for 10 devices the cost remains below 900 ms. These are very promising results, as they support the idea of frequent key updates for enhanced security.

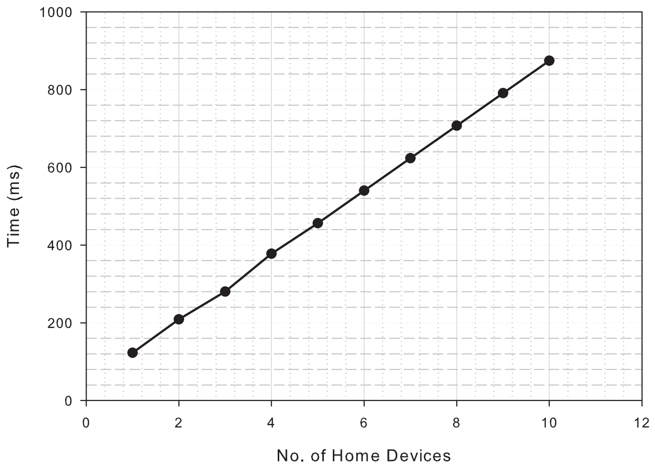


Fig. 6. Response time for distributed IBC key generation

Next, we tested the scalability of the authenticated key exchange protocol, and the results are shown in Fig. 7. The solid line corresponds to the response time at the admin device for exchanging session keys with n IoT devices. The almost

[3]https://crypto.stanford.edu/pbc/
[4]https://gmplib.org/
[5]https://www.openssl.org/

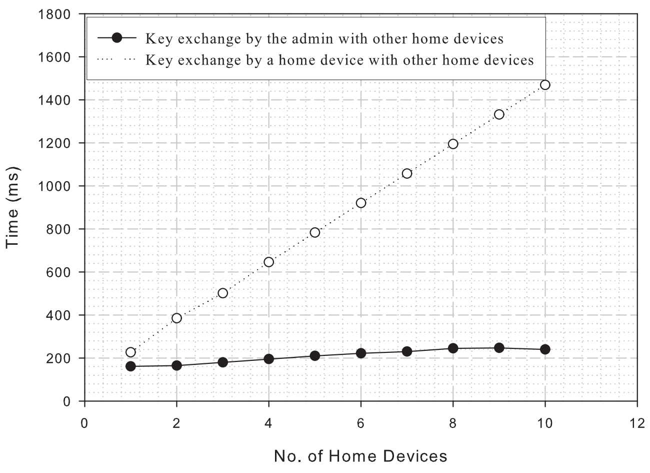


Fig. 7. Response time for authenticated key exchange

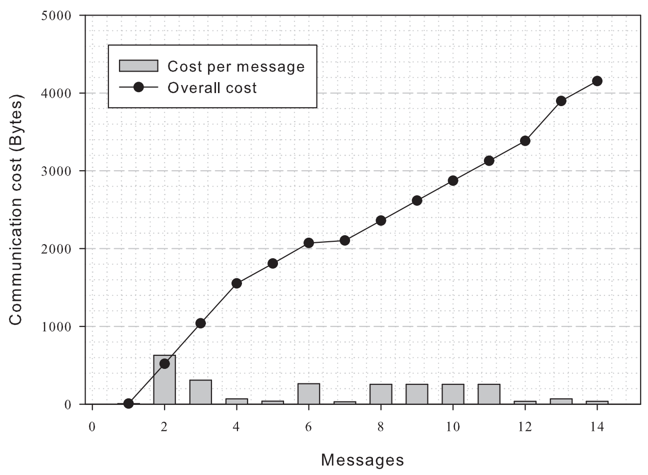


Fig. 8. Communication cost for one device invoking all three protocols with the admin device

constant cost stems from the large performance gap in the pairing computations for the two types of devices (9 ms vs. 82 ms). As such, the admin device is able to compute all session keys in approximately the same time it takes all other devices to compute one key. On the other hand, the dotted line illustrates the same cost for the case of an IoT device. Naturally, the response time grows linearly with the network size, due to the homogeneity of the devices. Note that, the response time remains under 1 sec, even for authenticated key exchanges between 6 devices. This is a very nice result that encourages frequent key updates. Furthermore, as we discussed previously, we do not expect a smart home device to establish short-term session keys with a large number of other devices.

Finally, Fig. 8 illustrates the communication cost for a single IoT device that invokes all three protocols in succession with the admin device (these are the only two devices in the network). It shows the cost from the moment the device starts the registration process, until it establishes a session key with the admin device. There is a total of 14 messages exchanged, whose sizes are represented with vertical bars (the solid line indicates the cumulative cost). Clearly, the communication overhead is very low, requiring just 4 KB of data transmissions. For n devices, the cost would increase linearly, as it involves the exchange of the same type of messages per added device.

VII. Conclusions

Security is a vital requirement for every smart home system. But any security solution must also take into account the resource limitations of the underlying devices. To this end, we proposed an identity-based security system that addresses the computational and storage limitations of IoT devices. Our protocols simplify the pairwise authentication and key exchange among smart home devices, without the need for a trusted and centralized key generation center. In addition, they are secure against standard network attacks, including man-in-the-middle, impersonation, and replay attacks. We implemented all protocols on Raspberry Pi 3 devices and experimentally evaluated their performance on a WiFi-based network. The results indicate that the proposed smart home security solution is very efficient and scales well with the home network size.

References

[1] D. J. Malan, M. Welsh, and M. D. Smith, "A public-key infrastructure for key distribution in tinyos based on elliptic curve cryptography," in *Sensor and Ad Hoc Communications and Networks, 2004. IEEE SECON 2004. 2004 First Annual IEEE Communications Society Conference on*. IEEE, 2004, pp. 71–80.

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

[3] L. Yang, C. Ding, and M. Wu, "Establishing authenticated pairwise key using pairing-based cryptography for sensor networks," in *2013 8th International Conference on Communications and Networking in China (CHINACOM)*. IEEE, 2013, pp. 517–522.

[4] P. Szczechowiak and M. Collier, "Tinyibe: Identity-based encryption for heterogeneous sensor networks," in *2009 International Conference on Intelligent Sensors, Sensor Networks and Information Processing (ISSNIP)*. IEEE, 2009, pp. 319–354.

[5] Y. Mao, J. Li, M.-R. Chen, J. Liu, C. Xie, and Y. Zhan, "Fully secure fuzzy identity-based encryption for secure iot communications," *Computer Standards & Interfaces*, vol. 44, pp. 117–121, 2016.

[6] H. Nicanfar, P. Jokar, and V. C. Leung, "Efficient authentication and key management for the home area network," in *Communications (ICC), 2012 IEEE International Conference on*. IEEE, 2012, pp. 878–882.

[7] R. H. Jacobsen, S. A. Mikkelsen, and N. H. Rasmussen, "Towards the use of pairing-based cryptography for resource-constrained home area networks," in *Digital System Design (DSD), 2015 Euromicro Conference on*. IEEE, 2015, pp. 233–240.

[8] D. Boneh and M. Franklin, "Identity-based encryption from the weil pairing," *SIAM journal on computing*, vol. 32, no. 3, pp. 586–615, 2003.

[9] ——, "Identity-based encryption from the weil pairing," in *Annual international cryptology conference*. Springer, 2001, pp. 213–229.

[10] R. Sakai and M. Kasahara, "Id based cryptosystems with pairing on elliptic curve." *IACR Cryptology ePrint Archive*, vol. 2003, p. 54, 2003.

[11] C. Gentry, "Practical identity-based encryption without random oracles," in *Annual International Conference on the Theory and Applications of Cryptographic Techniques*. Springer, 2006, pp. 445–464.

[12] C. Cocks, "An identity based encryption scheme based on quadratic residues," in *IMA International Conference on Cryptography and Coding*. Springer, 2001, pp. 360–363.

[13] L. Ducas, V. Lyubashevsky, and T. Prest, "Efficient identity-based encryption over ntru lattices," in *International Conference on the Theory and Application of Cryptology and Information Security*. Springer, 2014, pp. 22–41.

[14] S. McCarthy, N. Smyth, and E. O'Sullivan, "A practical implementation of identity-based encryption over ntru lattices," in *IMA International Conference on Cryptography and Coding*. Springer, 2017, pp. 227–246.

[15] G. De Meulenaer, F. Gosset, F.-X. Standaert, and O. Pereira, "On the energy cost of communication and cryptography in wireless sensor networks," in *2008 IEEE International Conference on Wireless and Mobile Computing, Networking and Communications*. IEEE, 2008, pp. 580–585.

[16] X. Yao, Z. Chen, and Y. Tian, "A lightweight attribute-based encryption scheme for the internet of things," *Future Generation Computer Systems*, vol. 49, pp. 104–112, 2015.

[17] H. Nicanfar, P. Jokar, K. Beznosov, and V. C. Leung, "Efficient authentication and key management mechanisms for smart grid communications," *IEEE systems journal*, vol. 8, no. 2, pp. 629–640, 2014.

[18] Q. Gao, "Biometric authentication in smart grid," in *Energy and Sustainability Conference (IESC), 2012 International*. IEEE, 2012, pp. 1–5.

[19] A. Shamir, "How to share a secret," *Communications of the ACM*, vol. 22, no. 11, pp. 612–613, 1979.

[20] R. Gennaro, S. Jarecki, H. Krawczyk, and T. Rabin, "Robust threshold dss signatures," in *International Conference on the Theory and Applications of Cryptographic Techniques*. Springer, 1996, pp. 354–371.

[21] R. Gennaro, S. Goldfeder, and A. Narayanan, "Threshold-optimal dsa/ecdsa signatures and an application to bitcoin wallet security," in *International Conference on Applied Cryptography and Network Security*. Springer, 2016, pp. 156–174.

[22] L. Nguyen, "Partially interactive threshold rsa signatures," *Oxford computing technical report*, 2005.

[23] J. C. Choon and J. H. Cheon, "An identity-based signature from gap diffie-hellman groups," in *International workshop on public key cryptography*. Springer, 2003, pp. 18–30.

[24] S. M. Bellovin and M. Merritt, "Encrypted key exchange: password-based protocols secure against dictionary attacks," in *1992 IEEE Computer Society Symposium on Research in Security and Privacy, Oakland, CA, USA, May 4-6, 1992*, 1992, pp. 72–84. [Online]. Available: https://doi.org/10.1109/RISP.1992.213269