# SecureMatch: Scalable Authentication and Key Relegation for IoT Using Physical-Layer Techniques

Hanif Rahbari
Department of Computing Security
Rochester Institute of Technology, Rochester, NY
Email: rahbari@mail.rit.edu

Jinshan Liu and Jung-Min (Jerry) Park
Bradley Department of Electrical and Computer Engineering
Virginia Tech, Arlington, VA
Email: {jinshan, jungmin}@vt.edu

***Abstract*—Conventional authentication approaches, whether based on (preconfigured) symmetric or asymmetric keys, cannot react timely to large-scale attacks on Internet of Things (IoT) devices because they need to manage (i.e., create, certify, distribute, and revoke) long lists of keys or device identities. In particular, existing key revocation schemes are inept at handling the extreme scale of IoT networks. In this paper, we propose a new concept called *relegation* to facilitate developing *SecureMatch*, a protocol to cope with scalability while incurring minimal time and/or space complexity. Relegation is like a fuzzy key revocation by which only the trustworthiness level of a vulnerable but not compromised device is reduced. Inspired by lightweight identity-based signatures, *SecureMatch* uses public features of a class of devices as their public keys to characterize and relegate all similar devices that share a common vulnerability. Further, the proposed protocol encodes those public features, which are often collected from different layers of the protocol stack, into a device *tag*, and then transmits this *tag* in the physical layer using a reliable preamble-embedding technique. *SecureMatch* has low overhead and enables an authenticator to quickly authenticate a set of features as long as the devices exhibit the valid credentials. The performance of *tag* extraction was evaluated through LabVIEW and USRP experiments using commodity devices as transmitters.**

***Index Terms*—Internet of Things (IoT), authentication, scalable relegation, frame preamble, PHY-layer, USRP.**

## I. Introduction

It is expected that the number of connected Internet of Things (IoT) devices worldwide will soon exceed 10 Billion; reaching 20 Billion by 2020 [1]. The pervasive deployment of IoT devices is expected to bring about faster move towards smarter systems in agriculture, healthcare, transportation, vehicle-to-vehicle (V2V) communications, etc. In particular, Juniper Research reports that by the year of 2022, $50\%$ of new vehicles will be shipped with V2V hardware to enable real-time peer-to-peer (P2P) communication between them [2]. Such a growth in the scale of emerging networks underscores the paramount need for a massively scalable *authentication* and *key revocation* scheme.

As the size of a network increases, it becomes increasingly difficult to authenticate every node. In the meanwhile, it becomes easier for adversaries to victimize and hijack vulnerable nodes to launch various attacks. For example, the recent potent Distributed Denial of Service (DDoS) attack against Internet performance management company *Dyn* was carried out by a larger network of Mirai botnets made up of up to $600,000$ compromised, but similar IoT devices [3]. This massive attack took down hundreds of web services, including Airbnb, Amazon, GitHub, Netflix, PayPal, Twitter, etc. To form the IoT botnet, Mirai primarily infected certain DVRs, IP cameras, and consumer routers [3], which obviously had common features. In another incident, *Flashpoint* researchers found more than $515,000$ Internet devices as of October 2016 that were vulnerable to the flaws they discovered [4]. Many of such mass-produced IoT devices may not be patchable after deployment and will remain vulnerable throughout their lifetime [5], [6], encouraging adversaries to continue launching similar attacks (e.g., BrickerBot attack in April 2017 [7]). Specifically, Symantec reports that vendors are hard-coding lifetime passwords even as easy as 'admin' and 'root' into devices without giving users the ability to change them [8].

The legacy notion of revocation is inept at handling the extreme scale of IoT networks because it pigeonholes a key or vulnerable device into one of two rigid states – valid or invalid. In contrast, we propose the fuzzy concept of *relegation* as the act of reducing the trustworthiness level of a device, which affords flexibility in how a vulnerable device's trustworthiness is gauged. For example, the impact of the Mirai attack could have been mitigated effectively if the Internet gateways and service providers were able to quickly react and perform large-scale device relegation to quarantine the traffic coming from other alike DVRs and cameras without revoking them.

In addition, it is not a scalable idea to rely on unique identities of thousands IoT devices, as what existing schemes offer, to populate the blacklist of vulnerable/compromised devices or the list of authorized ones. The maintenance cost of such lists in the cloud, and the overhead and the delay of publishing them to the gateways or resource-constrained IoT devices grow significantly as more IoT devices are activated. Relying on unique identities, such as, Android ID, MAC address, and RF fingerprints [9], will face additional challenges. They may not stay constant over time and can even be maliciously tampered with. For example, MAC address randomization is being increasingly adopted by modern operating systems (e.g., Android, iOS, etc.), making it almost impossible to keep track of every device's new MAC address. It is also insecure to rely on RF fingerprints because an adversary can quite easily impersonate the fingerprint of another device to launch masquerade attacks [10]. One may take advantage of Public Key

978-1-5386-4586-4/18/$31.00 ©2018 IEEE

Infrastructure (PKI) to counter such attacks by creating and certifying a public key for each identifier/fingerprint. However, PKI incurs high maintenance, storage, and communication cost in large-scale IoT networks.

In PKI, a Certificate Authority (CA) is often in charge of managing the public keys. It implies extra transmissions to or from the CA to share a node's bundle of credentials with other network entities. It also requires either storing at each device the public keys of all the potential peers, or a ubiquitous connection to the CA to acquire and verify a public key, which is not always possible (e.g., when peer vehicles in V2V are not in the transmission range of a Road Side Unit (RSU)).

Identity-based signature (IBS) [11] is an alternative public-key scheme that uses a device's public identity as its public key. IBS is certificate-free and has lower overhead than PKI because there is no need to rely on CA to acquire public keys and their certificates. So it also offers good support for P2P communications. However, IBS has a limitation that there is no method to efficiently revoke the keys of a large number of vulnerable devices in large-scale IoT networks, a critical performance bottleneck. This is also the case in the recently proposed *AoT* protocol [12] that uses IBS for a low-overhead authentication and attribute-based access control for IoT.

In this paper, we take advantage of a lightweight (i.e., pairing-free) IBS to develop an efficient and massively scalable authentication and key relegation scheme, called *Feature-Based Authentication* (FBA). We argue that relegation is better than revocation when dealing with thousands of cheap IoT devices because it does not forcefully revoke or fully authorize vulnerable but not compromised ones. FBA is different from IBS in the sense that it uses common *features* of the devices rather than their distinct identities to distinguish vulnerable devices of the same type and reduce the key relegation overhead. In other words, rather than verifying whether a certain identity is vulnerable, we check whether the device belongs to a *class* of thousands of vulnerable devices; making FBA a scalable authentication scheme. (It can still be used if a class contains a single device.) We then propose *SecureMatch* protocol, which employs FBA to authenticate or relegate devices based on a plurality of their public and perhaps common features.

Our selected set of features includes hardware or physical (PHY) layer features, brand, type, operating system, wireless protocol, etc. Such features are distributed across multiple layers, making it difficult to collect and authenticate them all at the same layer. In fact, an authenticator (the entity that is trying to classify and authenticate a device) even may not have the permission (e.g., `READ_PHONE_STATE` permission in Android [13]) or a compatible application-layer protocol to acquire certain features. To account for this challenge, a device in *SecureMatch* embeds a digest of its features, called the *tag*, in the PHY layer and sends it to the authenticator so as to facilitate authentication. As a proof of concept, we leverage the preamble-embedding technique in [14] to embed this *tag* (and perhaps its signature) in the frame preamble of the WiFi-based systems, enabling the receiver to instantly extract the *tag*. The technique in [14] also maintains backward compatibility with legacy devices and so can easily be integrated into existing IoT networks.

The main contributions of the paper are as follows:

- We propose *SecureMatch* protocol, which can (1) scale with the network size by distinguishing and relegating vulnerable devices based on their common features and (2) verify the authenticity of those features combined using lightweight FBA. The achieved scalability is in terms of storage, processing, and communication complexity.
- By transmitting the *tag*, a digest of carefully selected device features, using the frame preamble of WiFi systems, *SecureMatch* is able to facilitate and speed up the feature-based authentication by not going to upper layers.
- As a proof of concept, we implemented the *tag* extraction mechanism using USRP and commodity WiFi devices to verify the feasibility and efficiency of *SecureMatch*.

**Paper organization–** In Section II, we discuss the limitations of existing authentication schemes for IoT. We then present our adversary and system model in Section III and provide a high-level overview of the proposed *SecureMatch* scheme in Section IV. After that, we describe how to generate the *tags* based on devices' features in Section V. Details about our proposed FBA will be provided in Section VI along with a summary of the advantages of our scheme over existing ones. How to transmit *tags* in WiFi preamble and the corresponding experiment results will be shown in Section VII, before we conclude the paper in Section VIII.

## II. Existing Schemes and Their Limitations

Various authentication schemes have been proposed in the literature, some of which have already been implemented in real wireless systems. In the following, we discuss why such schemes do not adequately satisfy the requirements of emerging massive IoT networks.

**Device identification.** Several device *identification* schemes have been proposed (e.g., [9], [15], [16]) where identification is considered as a means for authentication. An authenticator needs to extract the supposedly unique features of a device (i.e., its fingerprint) to identify it. The performance of such schemes depends on how accurate a single device is identified among several *known* devices. That is, a classifier is trained using the samples previously extracted from the set of known devices [15]. It then exploits the distance between the (possibly inaccurate) extracted features of a device and the recorded features of the known devices for identification. For example, Brik *et al.* in [9] use normalized Euclidean distance and the authors in [16] compute a weighted combination of different features to measure the distance.

However, if underlying features are from PHY layer, estimation errors and automatic variations of these features over time (due to temperature and aging) may degrade the identification performance. If an estimated fingerprint of a device does not match its prerecorded fingerprint, it will give rise to the identification error rate. Even recent identification schemes, such as [16], fail to achieve a high success rate, mainly because of this fundamental limitation. They perform sufficiently well

only if the authenticator uses calibrated, high-end RF equipment (e.g., vector signal analyzer) for estimations. In contrast, our proposed approach does not depend on such equipment to meet acceptable performance requirements. This is compatible with the specifications of a typical IoT device, which is not equipped with high-end RF.

In addition, these schemes assume that the devices are honest and tamper-proof, i.e., the fingerprint of a device is not maliciously modified by an adversary. However, unless PUFs (Physically Unclonable Functions) are used, this assumption may not always hold, as shown in [10], and they need additional mechanisms to verify the authenticity of extracted features. The recent work by Guo *et al.* [17] proposed using Mahalanobis distance in a public-key encryption scheme to prevent spoofing biometric features for decrypting a message. However, their scheme relies on pairing operations, which are very expensive in resource-constrained IoT devices. It also requires storing multiple keys per device and transmitting several ciphertexts between the two parties so as to facilitate encryption (it is not a signature scheme). Furthermore, no key revocation protocol is discussed in this paper.

**Device-type identification.** Miettinen *et al.* [18] recently proposed a technique to identify the *type* of a WiFi device (rather than its unique fingerprint) in a small IoT network to enforce mitigation measures for those devices that have security vulnerabilities. In this technique, the access point (AP) collects the features of a device across the protocol stack (from link layer to application layer) and then identifies its type using machine learning-based classification. However, this technique enables only the AP to identify or authenticate a device. A device in this scheme has to obtain device-specific credentials from the AP to encrypt its traffic to the AP. Although relying on unique keys will curb the ability of a compromised device to eavesdrop on the traffic of other devices, it prevents a device from decrypting the traffic of another device and extracting its features. Hence, this architecture cannot be deployed in P2P scenarios, such as V2V. Furthermore, the authors assume that the devices are honest w.r.t. their features (i.e., they may have vulnerabilities but have not been compromised) when they obtain the credentials. In other words, they do not consider adversaries that attempt to carry out masquerade attacks by placing fraudulent "cloned" devices in a network.

**IoT-specific authentication schemes.** As an IoT device is usually miniaturized and constrained in resources, and so cannot perform sophisticated cryptographic operations that are common in more capable devices, various lightweight authentication schemes have been developed for IoT. The Constrained Application Protocol (CoAP) defined in RFC 7252 [19] is one of these schemes that defines authentication modes at the application layer based on pre-shared symmetric or asymmetric keys [20]. Although lightweight, CoAP and other similar schemes, such as, MQTT [21], are implemented at upper-layers of the protocol stack and so are oblivious to the hardware, PHY-layer, and MAC layer features. As discussed above, in many cases a combination of features from different layers is used to authenticate a device.

Key distribution and revocation are other limitations of these schemes. For example, CoAP supports pre-shared symmetric key mode. With billions of IoT devices, storing and using symmetric keys is unscalable. Although asymmetric key mode in CoAP can mitigate the storage issue, distributing, updating, or revoking them by the CA will be costly in large-scale IoT networks. *AoT* protocol [12] alleviates the key distribution problem by using IBS, but for revocation it just waits for the expiration of a compromised key, leaving it valid until then.

## III. System and Adversary Model

We consider two network setups: (1) a typical IoT network in homes and offices, where many devices are connected to one of multiple wireless gateway routers that are managed by a Trusted Third Party (TTP), (2) a V2V network where the vehicles have intermittent connection to RSUs or LTE towers that are managed by a TTP. The TTP is Internet-enabled and is being updated with the common features of vulnerable or blacklisted devices, once detected. In the following, we assume a network consists of an authenticator, a supplicant, the TTP, and an adversary. The authenticator can be the same device that will eventually receive the supplicant's message. But it can also be a intrusion detector or a blind receiver [22], who is not the intended receiver of the supplicant's message. In the later case, the detector only concerns about whether a certain IoT device is legitimate, and do not need to exchange messages with it. *We assume that the devices use the PHY-layer of WiFi.*

We also assume that the TTP can obtain a device's true features as long as the device is physically accessible. For example, if the human user manually resets the device to its factory settings, then the TTP can measure its true features (provided that the firmware is tamper-proof). Once an IoT device joins the network, the TTP obtains its features and determines whether or not they characterize a vulnerable device. The TTP conveys the public master key and a private key to a device only once when it joins the network (and its authenticity is verified by the TTP).

The adversary (or a compromised device) can bypass the firewall and masquerade any authentic device, and can overhear the communications between the authenticator and the supplicant, but not the communications to or from the TTP. Different from existing authentication schemes, we further assume that an adversary can take over a vulnerable device and masquerade its features whenever they are supposed to be acquired through a wireless connection. The adversary's goal is to exploit a vulnerability in several IoT devices to generate malicious traffic or access the network assets.

## IV. *SecureMatch* in a Nutshell

We now introduce our *SecureMatch* protocol at high level. The details will be discussed in the subsequent sections.

When a new IoT device wants to join a network, it needs to first register with the TTP. The TTP is involved only in the registration phase, and not in the actual authentication protocol. It generates and assigns a *tag* based on the new device features. To extract the features, the TTP may use

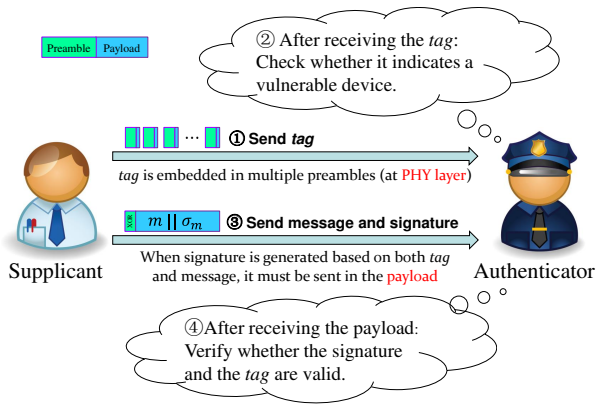



Fig. 1. The four main steps for authentication in *SecureMatch*.

`tcpdump` or other common feature extraction techniques. The *tag* will be used as the device's public key and will be transmitted in the preambles of a series of back-to-back frames. Next, the TTP will use its master secret key to generate a unique private key corresponding to this *tag*. Note that except TTP, no other user has the ability to generate the private keys.

Each time a class of vulnerable devices is published, the TTP will broadcast the *common* features that characterize perhaps thousands of such devices. Each authenticator maintains a *Feature Relegation List* (FRL) to store these common features.

When a supplicant wants to send a message, the *SecureMatch* protocol outlines the following steps (see Fig. 1):

1) The supplicant slices and embeds its *tag* in the preambles of a sequence of very short packets (one slice in each preamble) to initiate the process at the authenticator. The rationale behind transmitting the *tag* at the PHY-layer is its better speed and practicality (see Section VII).
2) Upon receiving those packets and without decoding their payload, the authenticator quickly extracts the *tag* and checks whether it contains the features of a vulnerable device. If it is a match, the authenticator will quarantine the subsequent packets of that supplicant, relegate its key, and report this incident to the gateway for further instructions. (The trustworthiness level to which the device is relegated can depend on the application.) Otherwise, it will wait for the subsequent packet(s) to check the authenticity of the *tag* against masquerade attacks (see Section VI).
3) The supplicant will transmit in the frame payload a message as well as a signature (which is generated based on both *tag* and the message) to the authenticator. The preamble of this frame contains the XOR of *tag* slices. In the occasions when the authenticator only wants to verify the supplicant's *tag* and does not intend to receive its message, the supplicant may instead embed the signature of the *tag* in preambles; expediting the overall process.
4) Once the remaining frames are received, the authenticator will extract and verify the signature.

Our *SecureMatch* scheme has the following properties:

- The authenticator obtains the supplicant's public key (the *tag*) directly from the supplicant itself. There is no need to query a third party for the supplicant's public key or its certificate. As a byproduct, it can support fast authentication in P2P communications because the TTP does not need to be involved during the process.
- The authenticator can detect if the *tag* of a compromised supplicant has been manipulated to evade from being characterized as vulnerable or compromised. Its authenticity is verified through verifying its signature. A *tag* is authentic only if it is consistent with its associated private key and the actual features. The private key is based on true features and can be generated only by the TTP.
- A class of vulnerable devices can be efficiently relegated without needing to retrieve a potentially long list of their identities. With relegation, the gateway puts such devices under surveillance or extra protection (e.g., firewall, DMZ) instead of fully revoking access to them.

## V. Device Classification Using *tag*

In this section, we explain the generation of devices' *tag*s in *SecureMatch* as a means to effectively represent their features. A *tag* is composed of the following two main parts:

1) Common features: The first part represents features that may be common in a class of IoT devices, like protocol version, operating system, brand, etc. These features can be used to efficiently identify large numbers of vulnerable or compromised devices. Each time a certain type (class) of devices is characterized as vulnerable, authenticators only need to verify whether a supplicant's features would classify it as a member of that class.
2) Unique features: This part of the *tag* is primarily used to help different IoT devices that have the same common features get distinct private keys from the TTP. It can also be used to authenticate a single device.

We acknowledge that in our FBA scheme, the length of *tag* can be longer than an identity in traditional IBS. In traditional IBS, as long as a feature is unique to a device, it can be used as *tag*. In other words, the unique features part of the *tag* can be sufficient for IBS. However, the longer length of *tag* is not significant, especially with respect to to the resulting overhead reduction in relegating keys of compromised devices in existing protocols. We will discuss the rough size of *tag* in this section, and compare the communication and computation complexity for different key revocation methods in section VI.

### A. IoT Device Classification

Classification of IoT devices plays a crucial role in *SecureMatch*. If we classify IoT devices into several classes of small sizes, the size of the FRL can grow quickly because each time a security threat may impact multiple classes of devices. The most extreme case is when each device forms a single class, and our scheme degenerates to the conventional Certificate Revocation List (CRL) schemes with a list of all compromised keys. On the contrary, if we classify IoT devices too broadly, each time vulnerable features are published, keys of a large number of safe devices may also be relegated, giving rise to false dismissal rate.

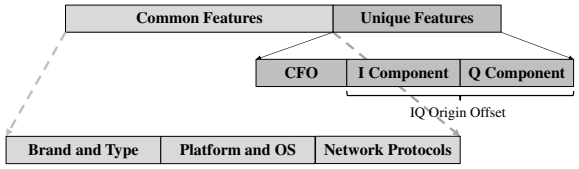



Fig. 2. Structure of a *tag*. All features are coded as bit strings and concatenated sequentially. Common features are used for classification and relegation, and unique features are used for generating unique private keys.

The encoding of the features into the *tag* in *SecureMatch* is similar to header generation in OSI layers. It first uses several bits to represent the first common feature, next few bits to represent the second one, and so on, as shown in Fig. 2. Such a linear coding may not be optimal because it does not minimize (compress) the number of coded bits, but is very light and straightforward for the authenticator to read all features. There is no need for complicated decoding and mapping scheme. It also makes it easy for a TTP to modify the coding scheme if new features are added, or outdated features are removed.

### B. Common Features

We leverage the findings of a recent IoT developer survey [23] as well as other studies on infected IoT devices (e.g., [3], [24], [25]) to compile a list of common features that identify vulnerable devices. In the following, the number in each bracket represents how many types are reported in [23] or approximated based on our observations.

**Brand and type–** As discussed in [3], [24], [25], a security flaw can be common among the devices with the same brand and similar functionalities. Because it is impractical to exhaust the list of all possible brands and types, we first consider the popular ones that are found in various IoT networks and then reserve a few additional placeholders; enabling the TTP to add more brands/types for a given network, if needed.

- Brand (64): Amazon, Samsung, Dahua, Philips, etc.
- Type (128): camera, DVR, router, lamp, thermostat, etc.
- Application domain (22): smart home, industrial automation, healthcare, agriculture, etc.
- Function (10): sensor, actuator, edge node, hub, etc.

**Platform and operating system (OS)–** A virus or hacking technique often targets a particular OS or cloud service [26]. For example, a Windows virus will not usually infect Linux.

- OS (100): Linux, Windows, FreeRTOS, etc. For open-source OSs, we take the distribution into account. For example, Linux has distributions Raspbian, Ubuntu, etc. For commercial OSs, we take the version into account, like Windows 7, 8, 10, etc. From [23], the number of popular OS distributions and versions is less than 100.
- Cloud service (11 [3]): Amazon AWS, Microsoft Azure, IBM Bluemix, Google cloud platform, etc.
- Hardware architecture (15): Atmel [24], ARM Cortex M7, Intel X86-64, 16-bit MCU, etc.

**Network protocols–** Several vulnerabilities have been discovered in wireless protocols used by IoT devices [24], [25]. Similarly, vulnerable devices can be identified based on their protocol banners (e.g., in Mirai botnet [3]).

- Application layer protocol (8): FTP, SSH, Telnet, etc.
- Messaging protocol (11): MQTT, XMPP, CoAP, etc.
- Wireless protocol and version (32): ZigBee [24], Z-Wave, 802.11n/ac/ad/ax, Bluetooth v4.0 (BLE), etc.

### C. Unique Features

To facilitate generating unique private key for each device, we look for features that are time-invariant, or at least remain the same for a reasonably long time. One may consider MAC address, IP address, Android ID, etc. as unique features. However, as discussed earlier, even MAC address may randomly change over time in new operating systems. Hence, we seek other features that are inherent to devices themselves.

As discussed in [15], several PHY-layer features remain constant, and can be used as device fingerprints. Even for distinct devices that come from the same manufacture, those features are often different [16]. In our scheme, we select the following two easily measured features to be the unique ones:

- **Carrier Frequency Offset (CFO)–** The difference between the operating frequency of the authenticator and the supplicant. CFO is specific to a pair of devices, and may change if a different pair is considered. So in our scheme, we let the supplicant compute its CFO with respect to the TTP as the global reference and add it to its *tag*. The authenticator can acquire the supplicant's CFO using its CFOs to the supplicant and to the TTP. The latter can be measured and stored once during the registration phase.
- **I/Q origin offset–** The offset of the origin of the received symbols compared to the ideal I/Q plane, expressed in I and Q components.

Theoretically, no two devices should have exactly the same PHY-layer features. But in practice, we need to use discrete finite values to represent those continuous values (i.e., encode PHY-layer features to a binary sequence). As a result, several devices may obtain the same *tag* after quantization. We add a few padding bits for better classification, in case multiple devices end up having exactly the same *tag*.

### D. Size of the tag

Since a *tag* needs to be embedded in multiple preambles (one preamble per frame), its size will influence how many frames to be transmitted, and hence, the authentication delay. Note that in practice, the features set can be network-specific, so the size of *tag*s will also be network-specific.

To approximate the size of a *tag*, we first consider the number of possible values for the common features (see Section V-A).

$$\lceil \log_2 64 \rceil + \lceil \log_2 128 \rceil + \cdots + \lceil \log_2 32 \rceil = 49 \text{ bits}. \quad (1)$$

in which $\lceil x \rceil$ means the smallest integer greater or equal to $x$.

In the unique features, CFO can often be estimated fairly accurately [16]. But to account for its estimation errors, we heuristically assign no more than 10 bits to represent it. I/Q

origin offset is usually estimated less accurately, so we assign 5 bits for both I and Q components. We also consider another 5 random bits for padding, in case two or more devices end up having the same *tag*. So the common and unique features combined make the whole *tag* be approximately 74 bits.

## VI. Feature-Based Authentication (FBA)

The most important component of *SecureMatch* is FBA, which is basically the same as IBS but with the *tag* used as the identity to make key relegation scalable. In this section, we first discuss the main advantage of IBS as well as the limitations of the existing key revocation protocols when IBS is used. Then we introduce FBA, which inherits the core structure of IBS schemes. We also show the advantages of *SecureMatch* by comparing communication and computation complexity in the authentication and relegation process, as well as the storage to maintain the FRL, with existing revocation protocols.

### A. Motivation

IBS eliminates the necessity for querying the CA (or TTP) to verify the authenticity of the public keys by having a public identity act as the device's *explicit* public key, obviating the need for certificates and their associated *implicit* public keys. So communication overhead and computation cost are significantly reduced if IBS is used. This property further facilitates P2P communication because it reduces the dependency on the TTP. In IBS, a device queries the TTP to receive a private key based on its identity/*tag* once only when it joins the network. Unlike PKI, only the TTP is able to generate private keys and other devices cannot generate one for themselves.

Although using IBS reduces the overhead in distributing the public keys and their certificates, key revocation becomes challenging. It is challenging because of not querying the TTP about the validity of a public key in each authentication process. Instead, the TTP has to either (a) change the master public key and reissue private keys for all the devices, or (b) notify all the users about the identity/*tag* of the compromised devices. Obviously, changing the master key means rebuilding the whole network, which is not practical. Hence in practice, the other approach is more commonly used.

There are two widely used protocols to make users aware of compromised devices, namely CRL and Online Certificate Status Protocol (OCSP) [27]. Several existing protocols, such as CoAP [19], use CRL. It is a list of revoked devices that have been issued and subsequently revoked by the TTP. Each time a device is revoked, it needs to be broadcasted and all devices will have to verify the authenticity of the new list and update their CRLs. Hence, every CRL may need to be updated once for each compromised device, which burdens network devices and bandwidth with non-production traffic. On the other hand, the number of compromised devices can be very large in large-scale IoT networks. Because of the limited storage of an IoT device, it may not be able to maintain such a large CRL.

OCSP addresses the shortcomings of CRL by having the authenticator query the TTP (or OCSP server) to check whether a supplicant has a valid key corresponding to its *tag*. The TTP responds with a signed message. So in OCSP, there is no need to maintain a list of compromised devices but the TTP involvement is necessary, preventing P2P communications. Furthermore, in terms of communication and computation complexity, the authenticator needs to receive and verify two signatures: one from the supplicant, and one from the TTP.

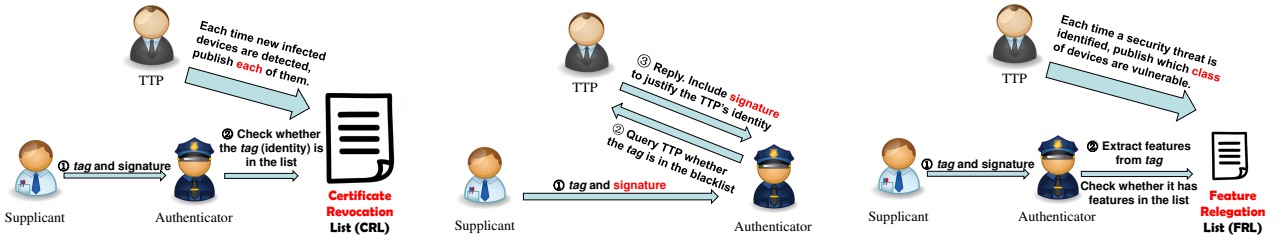
So CRL is impractical for key revocation in large-size IoT networks, while OCSP is expensive and cannot support P2P communications. To overcome these shortcomings, we propose the concept of FBA. Different from IBS, the features (coded into the *tag*) are used in FBA to check whether the supplicant belongs to a class of vulnerable devices. Because a vulnerable device that has those features may not have been compromised yet, we use relegation instead of revocation to allow that device continue to operate but under surveillance. The FRL stores only the common features of the vulnerable devices, significantly reducing the storage requirements because the number of common features is far less than the number of devices. Each time a class of vulnerable devices is detected, the TTP publishes their common features and all devices update their FRLs.

Fig. 3 illustrates the key revocation/relegation procedures of CRL, OCSP, and FRL. A comparison among these protocols as well as *AoT* protocol [12] is shown in Table I. *AoT* suggests renewing devices' private keys periodically, so it cannot take actions timely in case devices or keys are compromised, unless it uses CRL or OCSP. For a more concise notation, let $N_C$ denote the number of compromised devices and $N_F$ the number of classes of compromised devices. It is reasonable to assume $N_F \ll N_C$ in practical situations (e.g., only nine vendors were responsible for $500,000$ Mirai-infected devices [3]).

### B. Framework of Proposed FBA

Other than replacing identity with *tag*, FBA has the same framework as IBS. In FBA, the supplicant sends its *tag*, message and the signature to the authenticator. The authenticator first verifies the validity of the *tag*, then checks whether the signature is valid based on the *tag* and the received message.

For IoT devices with limited computational and memory resources, the underlying cryptography scheme should be lightweight. Because of the very high complexity in pairing operations, it is highly preferred to employ a scheme which does not use pairing. Our proposed FBA can employ any of the existing pairing-free IBS schemes, such as [28], [29], [30], [31]. The first ECC (Elliptic Curve Cryptosystem)-based IBS scheme, called BNN-IBS, was introduced in the first version of [28]. To send an ECC point $Q = (x, y)$, a sender usually only needs to send the $x-$component and the receiver is expected to compute $y-$component by solving a quadric equation $y^2 = x^2+ax+b$. However, solving this equation may require intensive modulo exponential operations, which can be time consuming for limited-resource IoT devices. Instead, the authors in [29] suggest sending both coordinates from supplicant to authenticator to reduce the overall complexity. They also propose a modified version of BNN-IBS, called vBNN-IBS, which reduces the signature size if both coordinates are




(a) Key revocation using CRL. It maintains the list of all compromised (and vulnerable) devices, which can be very large.

(b) Key revocation using OCSP. The TTP is involved during authentication, so no storage is needed to store the list of revoked devices.

(c) Key relegation using FRL (*SecureMatch*). It stores only the common features of vulnerable devices, so will not be large.

Fig. 3. Illustration of three key revocation/relegation schemes.

TABLE I
COMPARISON OF THREE DIFFERENT KEY REVOCATION/RELEGATION SCHEMES.

| Scheme | Signature verification overhead | Storage of revocation list | Overhead in updating revocation list | Support P2P |
|---|---|---|---|---|
| CRL | One from supplicant | $\mathcal{O}(N_C)$ | $\mathcal{O}(N_C)$ | Yes |
| OCSP [27] | One from supplicant+One from TTP | No such list | No such list | No |
| *AoT* [12] | One from supplicant | Should either use CRL or OCSP to detect compromised devices/keys timely | | |
| FRL (*SecureMatch*) | One signature from supplicant | $\mathcal{O}(N_F)$ | $\mathcal{O}(N_F)$ | Yes |

transmitted. So we adopt vBNN-IBS as our underlying IBS scheme, which is composed of the following four probabilistic polynomial algorithms $(\mathcal{G}, \mathcal{E}, \mathcal{S}, \mathcal{V})$:

- Parameter generation algorithm $\mathcal{G}$, which takes as input the security parameter $\eta$, and outputs the master public key $\mathsf{mpk}$ and master secret key $\mathsf{msk}$. Given parameter $\eta$, $\mathcal{G}$ takes the following steps:
  - Specify $E(\mathbb{F}_q)$, which is the group of points formed by an elliptic curve $E$ over a prime finite field $\mathbb{F}_q$.
  - Specify a prime $p$ in range $[2^\eta, 2^{\eta+1}]$ with $p^2$ not divided by the order of $E(\mathbb{F}_q)$.
  - Choose a base point $P$ in the elliptic curve of order $p$.
  - Select master secret key $x$ at random from $\mathbb{Z}_p$, and compute master public key $P_0 = xP$.
  - Choose two cryptographic hash functions $H_1 : \{0,1\}^* \to \mathbb{Z}_p$ and $H_2 : \{0,1\}^* \to \mathbb{Z}_p$.
  - Publish ($\mathsf{mpk}$, $\mathsf{msk}$)$= ((E(\mathbb{F}_q), P, p, P_0, H_1, H_2), x)$
- Key-generation algorithm $\mathcal{E}$, which takes input $\mathsf{msk}$ and $\mathsf{mpk}$, and for each *tag*, it outputs a private key $\mathsf{sk}_{\mathsf{tag}}$ corresponding to the user with this *tag*. It first picks a random number $r$ uniformly in $\mathbb{Z}_q$, and computes $R = rP$. It then uses master secret key $x$ to compute $s = r + H_1(\mathsf{tag}\|R) \cdot x$, where $\|$ denotes concatenation operator. The private key for this *tag* is set to be $\mathsf{sk}_{\mathsf{tag}} = (R, s)$.
- The signing algorithm $\mathcal{S}$, which takes as input parameters $\mathsf{mpk}$, $\mathsf{sk}_{\mathsf{tag}}$, and a message $m$, then outputs signature $\sigma_m$. It selects a random number $y$ uniformly in $\mathbb{Z}_q$, and computes $Y = yP$. Then it computes $h = H_2(\mathsf{tag}\|m\|R\|Y)$ and $z = y + hs$. The supplicant's signature $\sigma_m$ for message $m$ is the tuple $(R, h, z)$.
- The verification algorithm $\mathcal{V}$, which takes as input parameters $\mathsf{mpk}$, signature $\sigma_m$, message $m$ and the supplicant's *tag*, and determines whether $\sigma_m$ is a valid signature of $m$. It proceeds as follows. It outputs whether or not the equation $h = H_2(\mathsf{tag}\|m\|R\|(zP - h(R + cP_0)))$ holds, where $c = H_1(\mathsf{tag}\|R)$. If this equation holds, the signature (or identity) can be verified. Otherwise, simply reject this authentication.

Depending on IoT devices' resources, we can choose other IBS schemes. The IBS scheme in [30] combines symmetric and asymmetric cryptography, and tends to reduce computational complexity. But in order for authenticator to authenticate supplicant, the supplicant also needs to know the authenticator's public key, since the signature is generated by using both public keys. In addition, unlike traditional IBS which uses devices' *tags* as public keys, this scheme needs to generate public keys based on *tags*. Consequently, there is one additional step to verify the validity of public keys.

If the authenticator just wants to authenticate the *tag* itself, like an intrusion detector, the actual message $m$ is not important. Since the *tag* and its signature are embedded in preamble in this scenario, we can simply use a pre-defined bit stream as message $m$, or just leave $m$ empty, avoiding to send $m$ for each authentication and reducing the communication complexity. Note that it is robust against replay attack, because a new random number $y$ in signing algorithm $\mathcal{S}$ is generated each time. If the authenticator needs to authenticate the integrity of message $m$, only the *tag* is embedded in preamble. The message is still sent via the frame payload.

### C. Signature Size (Communication Complexity)

As highlighted by National Institute of Standards and Technology (NIST) [32], the security against key-recovery attacks should be at least 112 bits for IoT. So in our case, $|p|$ ($|p|$ is the number of bits to represent $p$) and $|q|$ should be at least 224 bits, because for a size of $\eta$ bits, the general birthday attack are expected to find collisions in $2^{\eta/2}$ steps. Similar analysis can be very easily extended to other security levels. If the authenticator has the ability to compute $y$-coordinate based on $x$, the signature size is $2|p| + |q| + 1 = 673$ bits (the extra one bit is used for indicating it is $+y$ or $-y$). If

the authenticator is not able to compute $y$-coordinate given $x$, both coordinates should be sent, and the signature size will be $2|p| + 2|q| = 896$ bits.

## VII. Embedding *tag* in the Preamble

Transmitting the *tag* (and its signature) at the PHY-layer has two advantages. First, any authenticator can instantly check all the features and does not need to wait for decoding and decrypting the payload to extract upper-layer ones. Note that in contrast to the payload of upper-layer protocols, the PHY-layer frame is never encrypted. Additionally, an authenticator may not always have the permission (e.g., `READ_PHONE_STATE` permission in Android [13]) to access certain system features. Second, a PHY-layer approach can be applied across all the devices with the same PHY-layer protocol, irrespective of their upper-layer protocols. That means a (blind) receiver does not need to know which transport or application layer protocol a device uses before it can check supplicant's features.

As a proof of concept, *SecureMatch* employs *P-modulation* [14], a PHY-layer message embedding technique, to communicate the *tag* and its signature (if the authenticator is not intended to receive the payload) at the PHY-layer. It uses the preamble of OFDM-based WiFi systems for embedding in such a way that unaware/legacy WiFi receivers who do not know *P-modulation* can operate as normal and receive the frames transmitted by a *P-modulation*-enabled device. That is, the technique is backward-compatible and does not need to forcefully modify existing communications and networking protocols. Hence, *SecureMatch*-enabled devices do not need to form a separate network for their own and can easily be integrated into existing IoT networks. *P-modulation* is also robust to mobility because of short duration of the preamble. Hence, it can also be adopted for V2V communications with small coherence times.

Compared to other PHY-layer embedding techniques such as FEAT [22], *P-modulation* is able to reliably embed 5 bits or more in the preamble of each frame. FEAT scheme can embed only 1 bit per frame. The actual embedding capacity of *P-modulation* depends on the underlying WiFi standard and the expected level of reliability, and can be higher than 5 bits per frame. According to [14], when the device is equipped with a single antenna (and uses IEEE 802.11a standard), it can embed 5–7 bits and match the bit-error-rate (BER) performance of BPSK modulation scheme, irrespective of the accuracy of CFO estimation. If the CFO estimation is accurate, it achieves BER as low as $10^{-4}$. The embedded bit sequence can be expanded to 8 at the expense of reduced reliability. On the other hand, if the device uses more recent 802.11n or 802.11ac standard for multiple-antenna systems, the capacity can be extended to 19–21 bits, depending on the expected reliability [14].

Besides the analytical performance comparisons in Table I, in the following we evaluate the performance of the *tag* extraction technique using a commodity WiFi NIC. The authors in [14] used only USRPs, i.e., one USRP as the transmitter and one as the receiver, to evaluate *P-modulation* in a controlled experimentation setup (i.e., with accurate CFO estimation) and over the narrow bandwidth of 1 MHz. In contrast, in here we consider a WiFi NIC as the transmitter and a USRP as the receiver to capture real WiFi transmissions over the 20 MHz bandwidth air interface (see Fig. 4). The specifications of the devices are shown in Table II. The transmitter runs IPERF with very short packet sizes of 8 bytes. We implement only the demodulation component of *P-modulation* (with certain modification) as we cannot modify the firmware of the NIC to embed different bit sequences in the preamble. Hence, the transmitted *tag* (the bit sequence corresponding to the standardized preamble) is assumed to be constant and we evaluate the BER performance of extracting this *tag*.

TABLE II
Comparison of three different key revocation protocols

| Role | Wireless interface model | Standard |
|---|---|---|
| Transmitter (supplicant) | Intel®Dual Band Wireless-AC 8260 | 802.11ac |
| Receiver (authenticator) | NI USRP-2922 | 802.11a |

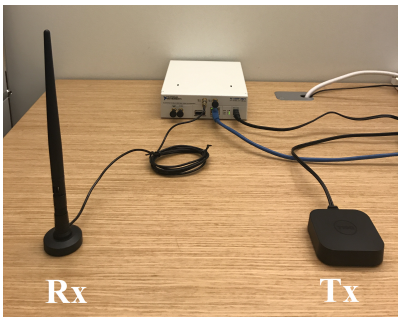



Fig. 4. Experiment setup (scenario #1).

We experiment the *tag* extraction under two different scenarios and show the results in Fig. 5. In scenario #1, we place the devices in a storage lab environment with line-of-sight. In scenario #2, we place the devices in an office environment with several objects around. The *tag*s are uncoded, i.e., no channel coding is used to improve the robustness of the embedded *tag* against channel impairments. The authors in [14] report that if CFO estimation is erroneous when using USRPs, the BER of any scheme, including BPSK and *P-modulation*, will be severely deteriorated. The results in Fig. 5 exhibit a similar trend as the CFO estimation is not accurate. Nevertheless, the results also show that the BER can be as low as 0.02 when 6 bits are embedded in a preamble. That means the entire 6-bit sequence will be correctly extracted with about 90% success rate. This rate increases to 99.8% if only 3 bits are embedded in each preamble. The performance can be improved if better CFO estimation mechanisms are employed.

In *SecureMatch* scheme, the embedded bit sequence is the supplicant's *tag*. From Fig. 5, we see that a relatively high reliability can be achieved when 5 bits are embedded in each preamble. In this case, if we only embed the *tag*, then $\lceil \frac{74}{5} \rceil = 15$ short packets are needed to be transmitted. If both the *tag* and its related signature are to be embedded in the preambles, then $\lceil \frac{672+74}{5} \rceil = 150$ packets are needed when we choose to send only one coordinate of ECC pints. Although we need to transmit several short packets in this case, it enables devices using incompatible upper-layer protocols to

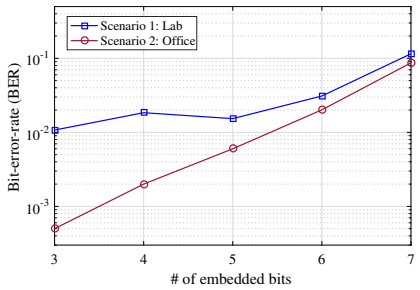



Fig. 5. BER performance of *tag* extraction versus different number of bits embedded in a frame preamble. The CFO estimation is not highly accurate, which creates a performance floor even when only 3 bits are embedded.

authenticate each other. To achieve a very high bit success rate, it is more advisable to embed only 3 bits per frame.

Alternatively, one may employ channel coding or retransmission techniques to account for lost or corrupted preambles. In fact, the XOR of the *tag* slices embedded in the preamble of the message frame can be considered for error detection and correction to some extent. However, the design of specific and stronger channel codes for *P-modulation* is beyond the scope of this paper and is left for future work.

## VIII. Conclusion

In this paper, we proposed *SecureMatch*, a scalable protocol that is also able to support P2P authentication in large-size IoT networks. Instead of storing the list of all compromised devices, the devices in *SecureMatch* only need to save the common features of vulnerable devices, significantly reducing the time and space complexity. Moreover, by relegating keys based on the class of devices (characterized by their common features), an authenticator has the flexibility of adjusting the trustworthiness of a vulnerable but not compromised without necessarily revoking access to it.

As a proof of concept, we also embed a digest of the features (the *tag*) in the preamble of WiFi systems. By doing so, we are able to speed up the authentication procedure by transmitting the *tag* without going to upper layer. Moreover, it can allow devices to authenticate each other without using the same upper-layer protocols. With USRP experiments, we demonstrate the practicality of such PHY-layer technology using a commodity transmitter.

## Acknowledgment

This work was supported in part by NSF, the industry affiliates of the Broadband Wireless Access & Applications Center (BWAC), and the Wireless@Virginia Tech group. Any opinions or conclusions expressed in this paper are those of the author(s), and do not necessarily reflect the views of NSF.

## References

[1] R. van der Meulen, "Gartner says 8.4 billion connected "things" will be in use in 2017, up 31% from 2016," https://goo.gl/X9bUr9, Feb. 2017.
[2] Juniper Research, "Consumer connected cars–applications, telematics & V2V 2017-2022," https://goo.gl/9x9amg, May 2017.
[3] M. Antonakakis *et al.*, "Understanding the Mirai botnet," in *Proc. 26th USENIX Security Symp.*, Vancouver, BC, 2017, pp. 1093–1110.
[4] Z. Wikholm, "When vulnerabilities travel downstream," https://goo.gl/iqKasw, Oct. 2016.
[5] KrebsonSecurity, "Hacked cameras, DVRs powered today's massive Internet outage," https://goo.gl/tUiDl0, Oct. 2016.
[6] T. Benson and B. Chandrasekaran, "Sounding the bell for improving Internet (of Things) security," in *Proc. Workshop Internet of Things Security and Privacy*, Dallas, Texas, USA, 2017, pp. 77–82.
[7] C. Kolias, G. Kambourakis, A. Stavrou, and J. Voas, "DDoS in the IoT: Mirai and other botnets," *Computer*, vol. 50, no. 7, pp. 80–84, Jul. 2017.
[8] D. Palmer, "Is 'admin' password leaving your IoT device vulnerable to cyberattacks?" https://goo.gl/Ej12MG, Apr. 2017.
[9] V. Brik, S. Banerjee, M. Gruteser, and S. Oh, "Wireless device identification with radiometric signatures," in *Proc. 14th ACM in. conf. Mobile Computing and Networking (MobiCom)*, 2008, pp. 116–127.
[10] B. Danev, H. Luecken, S. Capkun, and K. El Defrawy, "Attacks on physical-layer identification," in *Proc. third ACM conf. Wireless network security (WiSec)*, 2010, pp. 89–98.
[11] A. Shamir *et al.*, "Identity-based cryptosystems and signature schemes," in *Crypto*, vol. 84. Springer, 1984, pp. 47–53.
[12] A. L. M. Neto *et al.*, "AoT: Authentication and access control for the entire IoT device life-cycle," in *Proc. 14th ACM Conf. Embedded Network Sensor Systems (Sensys)*, Stanford, CA, USA, Nov. 2016.
[13] Z. Fang, W. Han, and Y. Li, "Permission based android security: Issues and countermeasures," *Comput. & Security*, vol. 43, pp. 205–218, 2014.
[14] H. Rahbari and M. Krunz, "Exploiting frame preamble waveforms to support new physical-layer functions in OFDM-based 802.11 systems," *IEEE Trans. Wireless Commun*, vol. 16, no. 6, pp. 3775–3786, Jun. 2017.
[15] Y. Shi and M. A. Jensen, "Improved radiometric identification of wireless devices using MIMO transmission," *IEEE Trans. Inf. Forensics Security*, vol. 6, no. 4, pp. 1346–1354, 2011.
[16] T. D. Vo-Huu, T. D. Vo-Huu, and G. Noubir, "Fingerprinting Wi-Fi devices using software defined radio," in *Proc. 9th ACM Conf. Security & Privacy in Wireless and Mobile Netw. (WiSec)*, Jul. 2016, pp. 3–14.
[17] F. Guo, W. Susilo, and Y. Mu, "Distance-based encryption: How to embed fuzziness in biometric-based encryption," *IEEE Trans. Inf. Forensics Security*, vol. 11, no. 2, pp. 247–257, 2016.
[18] M. Miettinen *et al.*, "IoT sentinel: Automated device-type identification for security enforcement in IoT," in *Proc. 37th IEEE Int. conf. Distributed Computing Systems (ICDCS)*, Jun. 2017, pp. 2177–2184.
[19] C. B. Z. Shelby, K. Hartke, "The constrained application protocol (CoAP)," Internet Requests for Comments, RFC 7252, Jun. 2014.
[20] J. Granjal, E. Monteiro, and J. S. Silva, "Security for the Internet of Things: a survey of existing protocols and open research issues," *IEEE Commun. Surveys Tuts.*, vol. 17, no. 3, pp. 1294–1312, 2015.
[21] A. Banks and R. Gupta, "MQTT 3.1.1, OASIS standard," 2014, 3.1.1., OASIS Standard.
[22] V. Kumar, J.-M. Park, and K. Bian, "Blind transmitter authentication for spectrum security and enforcement," in *Proc. ACM Conf. Comput. Commun. Security (CCS)*, Scottsdale, Arizona, USA, 2014, pp. 787–798.
[23] Eclipse IoT Working Group and others, "IEEE, Agile-IoT EU, and IoT council," 2017.
[24] E. Ronen, A. Shamir, A.-O. Weingarten, and C. O'Flynn, "IoT goes nuclear: Creating a ZigBee chain reaction," in *IEEE Symp. Security and Privacy (SP)*, San Jose, CA, USA, May 2017, pp. 195–212.
[25] E. Fernandes, J. Jung, and A. Prakash, "Security analysis of emerging smart home applications," in *IEEE Symp. Security and Privacy (SP)*, 2016, pp. 636–654.
[26] G. Leopold, "AWS cloud hacked by bitcoin miners," https://goo.gl/Q9kAMJ, Oct. 2017.
[27] S. Santesson *et al.*, "X.509 Internet public key infrastructure online certificate status protocol-OCSP," Tech. Rep., 2013.
[28] M. Bellare, C. Namprempre, and G. Neven, "Security proofs for identity-based identification and signature schemes," *J. of Cryptology*, vol. 22, no. 1, pp. 1–61, 2009.
[29] X. Cao, W. Kou, L. Dang, and B. Zhao, "IMBAS: Identity-based multi-user broadcast authentication in wireless sensor networks," *Comput. Commun.*, vol. 31, no. 4, pp. 659–667, 2008.
[30] K. T. Nguyen, N. Oualha, and M. Laurent, "Lightweight certificateless and provably-secure signcryptosystem for the Internet of Things," in *Proc. IEEE TrustCom/BigDataSE/ISPA*, vol. 1, Helsinki, Finland, Aug. 2015, pp. 467–474.
[31] S. Chatterjee, C. Kamath, and V. Kumar, "Galindo-Garcia identity-based signature revisited," in *Proc. Int. Conf. Information Security and Cryptology*, 2012, pp. 456–471.
[32] K. A. McKay, L. Bassham, M. S. Turan, and N. Mouha, "Report on lightweight cryptography," Tech. Rep., Mar. 2017, NISTIR 8114.