Cryptiny: Compacting Cryptography for Space-Restricted Channels and its Use-case for IoT-E2EE

Liron David
Department of Computer Science
Weizmann Institute of Science
and Google

Omer Berkman
Department of Computer Science
The Academic College Tel-Aviv
and Google

Avinatan Hassidim
Department of Computer Science
Bar Ilan University
and Google

David Lazarov
Google

Yossi Matias
Department of Computer Science
Tel-Aviv University
and Google

Moti Yung
Department of Computer Science
Columbia University
and Google

***Abstract*—We present a novel cryptographic paradigm denoted "cryptiny:" Employing a single cryptographic value for several security goals, thus "compacting" the communication sent over a space-restricted (narrow) channel, while still proving security. Cryptiny is contrary to the classical cryptographic convention of using a separate cryptographic element for each security goal.**

Demonstrating the importance of cryptiny, we employ it for securing a critical IoT configuration in which a broadcasting "thing" (called beacon) operates within stringent bandwidth constraints. In this setting, a compact BLE-broadcasting beacon lacking Internet connectivity efficiently directs brief (non fragmented) messages to its remotely pre-paired owner in real-time. Communication transpires through BLE-to-IP gateway devices denoted observers, (typically smartphones in the beacon's vicinity), and subsequently via a cloud app server. The gateway device as well, piggybacks on the transmission a secure and private message to the owner. This configuration is a generic setting for the current and future IoT real-time ecosystems, where billion of owners, beacons, and observers operate.

The configuration instances (analogous to TLS instances over the Internet) imposes high security and privacy demands. We prove that our cryptiny-based protocol for securing the above configuration achieves CCA-secrecy for the beacon's and the observer's messages with backward and forward security for the observer's message, as well simultaneously achieving mutual privacy for beacons and for observers. Achieving backward and forward security is important since beacon devices may be far from their owners for a long duration and may be passively tampered with. In addition, for the backward security proof we develop a new encryption scheme we call "shifted-DHIES" ("SDHIES" for short), which generalizes DHIES. An interesting feature of SDHIES is that encryption is performed with a function of the public key rather than the public key itself.

I. Introduction

As new communication configurations arise, novel cryptographic solutions are imperative to address their unique security requirements effectively. For example, SSL/TLS were developed for securing the Internet or more precisely the HTTP protocol, WPA/WPA2 were developed to secure WiFi, and end-to-end encryptions such as the Signal protocol were developed to secure mobile instant messaging.

In this paper we develop a new cryptographic paradigm we denote "cryptiny" which is motivated by and is especially useful in the area of Internet Of Things (IoT) in which the "things" (beacons) are extremely limited in their broadcast bandwidth. The idea of the cryptiny paradigm is to pack multiple security mechanisms in a single short cryptographic value.

Obviously, the cryptiny paradigm is contrary to the classical cryptographic convention of using a separate cryptographic element for each security goal. We note that there are works such as [6], [10] which re-use cryptographic keys in order to limit the number of keys. However, cryptiny is different since its goal is to decrease the cryptographic footprint in terms of its length (in our case, the beacon's broadcast length), and not in terms of the number of keys employed. Cryptiny may be a crucial primitive in applications where the length of cryptographic messages must be limited and retrofitted into already defined fields.

Specifically, we employ cryptiny for securing the following configuration type in the IoT which we denote the IoT-E2EE configuration (see Figure 1) which may be executed extensively among billion of players (driven by potentially every mobile phone owner and its beacons): The first player, a Bluetooth Low Energy (BLE) **beacon** broadcasts over BLE its identity and a short message we denote "signal." The second player is a BLE-to-IP gateway device denoted **observer** (usually a smartphone), which happens to be in the beacon's vicinity for a short time period. This observer forwards over IP the beacon's broadcasts along with its own message (e.g., the observer's location) to the third player, a **cloud server**. The cloud server forwards the beacon's signal and the observer's message to the fourth player, the **beacon's owner**.

This configuration is used, for example, in asset-tracking

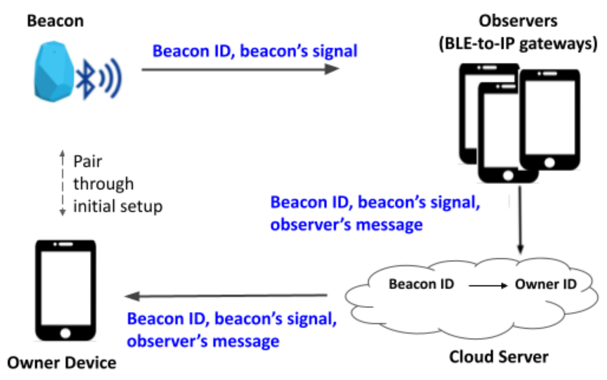
Beacon
Beacon ID, beacon's signal
Observers (BLE-to-IP gateways)
Pair through initial setup
Beacon ID, beacon's signal, observer's message
Beacon ID → Owner ID
Beacon ID, beacon's signal, observer's message
Owner Device
Cloud Server

Fig. 1: The IoT-E2EE configuration in IoT

applications to enable users to find their lost "things," where in general the things have no internet connection. In such a scenario, the BLE beacon is attached to the "would be lost" thing, the beacon's short signal (2-3 bits) may be the beacon's battery status (e.g., low, medium, high) and the observer's message is its GPS location. Since the observer is close to the beacon and thing, the observer's location is also the thing's location.

It is noteworthy that this configuration bears resemblance to a single TLS session. Just as numerous concurrent sessions exist among internet entities, our IoT setup also encompasses multiple simultaneous sessions between numerous owners and their respective beacons where these sessions rely on multiple observers serving as gateways.

Crucial constraints in the above configuration, are that a beacon's broadcast needs to be short (as defined by the BLE standards) and unfragmented: A short broadcast is required in order to preserve the beacons' battery, and an unfragmented broadcast is needed to allow even a single observer which is briefly present in the beacon's vicinity to receive the entire message reliably. These two constraints are fundamental for maintaining the operational efficacy and reliability of our configuration given its resource-constrained environment (characteristic of IoT broadcasting deployments).

Our aim is to develop a basic real-time protocol, foundational to IoT, that maintains the essential constraints of short and unfragmented beacon's broadcast while supporting the following security, privacy and integrity requirements tailored to the IoT-E2EE configuration within the large scale beacons ecosystem:

1) Security:
   - Beacon's signal end-to-end CCA-security.
   - Observer's message end-to-end CCA-security with backward and forward security with respect to the beacon's secrets. Backward and forward security protects future and past observer's messages (respectively) in case beacon secrets are exposed. This is important in our scenario since beacons may be far from their owners for extended time periods and are thus susceptible to physical penetration and exposure of beacons' secret keys. It is worth mentioning that while forward security can be readily achieved by applying one-way function on the beacons' secret keys, achieving backward security has traditionally relied on frequent key updates through re-pairing between beacons and their owners. Since a beacon could be isolated in the field for an extended duration, a frequent re-pairing is not feasible in our scenario and therefore a novel method is required.
2) Privacy:
   - Beacon's broadcast indistinguishability. Although we specify beacon's broadcast indistinguishability as a privacy requirement, it actually implies two more basic privacy requirements: (i) (Owner Anonymity), preventing the identification of an owner through their beacon's broadcast. Indeed, an observer is just an arbitrary smartphone which happens to be in the owner's beacon vicinity and therefore an owner and observer should remain anonymous to one another; and (ii) (Non-tracking), ensuring that different beacons cannot be distinguished based on their broadcasts.
3) Integrity:
   - Public key certification. Under a strong adversary one has to have assurance that a public key presented to it is the one decided upon in system key management initiation.
4) Autonomous Key Management: As we discuss below, we require that the existing infrastructure for communication allows for key management with no addition (of Certification Authorities, etc.). Hence, we require the pairing phase of the beacon and its owner to span a reliable key management layer.

We call this highly constrained problem, the IoT-E2EE problem. Notice that the above requirements should hold for the entire ecosystem, namely, for, say, billions of beacons, owners and observers and one cloud server (or server farm).

As required, an owner and its beacon can exchange initial keys during pairing. Owners can also exchange information with the cloud server during initialization (registration to the beacon app) to enable the cloud server correctly route incoming beacons' broadcasts. In contrast, the privacy requirement forbids observers from identifying or tracking owners (hence regular PKI keys are not allowed, and further owner to observer keys are not possible due to scale primarily and privacy constraints as well).

The beacon's broadcast in our IoT-E2EE configuration which is the starting point of the protocol is the only information available to the observer. The challenge is to achieve all the above security, privacy, integrity and key management requirements while maintaining a short and unfragmented beacon's broadcast. Thus, our IoT-E2EE problem exemplifies the critical role of the cryptiny paradigm utilized by the beacon. Indeed, the novel protocol we introduce, denoted "The Cryptiny IoT-E2EE Protocol," compacts all requisite cryptographic functions from the beacon's broadcast into a singular succinct "cryptiny value." We then prove that despite the beacon broadcasting solely a single cryptiny value and given the key management restrictions as well, the protocol fulfills all privacy and security requirements effectively and achieves correctness in support of real-time communication.

An independent contribution of this paper is a new encryption scheme, we denote "shifted-DHIES" or "SDHIES" in short, which is a generalization of the very useful DHIES system [2]. An interesting feature of our scheme is that encryption is performed with a function of the public key rather than the public key itself. We use the SDHIES scheme to prove CCA-security with backward and forward security of the observer's message. SDHIES is needed since in our protocol, the private key is a function of a DHIES private key and a value that can be controlled by the attacker. We strongly believe that SDHIES can be useful in other applications with similar constraints.

A. *Our Contributions*

In this paper, we:

1) Suggest "Cryptiny" as described above.
2) Present a real-world application in which cryptiny is important and necessary. Specifically, we specify and present the "IoT-E2EE problem," for the beacon ecosystem, under its key management and operational constraints.
3) Develop, in turn, the "Cryptiny IoT-E2EE Protocol."
4) Prove the security, privacy, and integrity properties of the Cryptiny IoT-E2EE Protocol.
5) Develop SDHIES, a generalization of the DHIES cryptographic scheme for our and for general use.

Finally, we note that the protocol presented here is part of a system under development in industry; it motivated the "Find My Device" functionality of Google, and will support its coming beacons' infrastructure.

II. Informal Description of the Constrains, our Protocol and its Security Guarantees

A. *The Short and unfragmented Beacon's Broadcast Constraints*

Given that the beacon is a small battery-powered device which may be in the field far from its owner for long time durations, frequent battery replacement is not practical. Therefore, to preserve beacons' battery, it is important to reduce the beacon's broadcast length. In fact, short broadcast is required by the beacon's standard. For example in Bluetooth Low Energy (BLE) the broadcast is length-limited (broadcast in BLE-4 and BLE-5 are limited to 37 and 256 bytes, respectively).

Furthermore, the beacon's message should be transmitted over a single broadcast and should not be fragmented into multiple broadcasts. This is since an observer may be moving along, and such a moving observer may not stay long enough in close proximity to the beacon to be able to read multiple broadcasts. We therefore insist that the beacon's broadcast be short and unfragmented.

These constraints on the one hand and the multiple security, privacy and integrity requirements on the other hand are the main reason for using the cryptiny paradigm. Indeed, to fulfill the aforementioned requirements, the short beacon's broadcast should encompass the following functional components: (1) CCA-secure encryption of the beacon's signal; (2) Certified cryptographic information that enables an observer to encrypt its message with backward and forward security; (3) serving also as an owner identifier. In addition, the beacon's broadcast must be ephemeral and pseudo-random.

Combining the above components into one short and unfragmented ephemeral pseudorandom broadcast was the main challenge of our protocol. To address this challenge, we leverage the cryptiny paradigm, compressing the cryptographic footprint of the beacon's broadcast into a single "cryptiny value" which is a point on an elliptic curve. The cryptiny value fulfills all the objectives outlined above.

B. *High Level Description of our Protocol*

Below we give a high level description of our protocol. In Section IV we provide a detailed description and motivations for the protocol and in Section V we present a formal cryptographic description.

1) A **beacon** broadcasts over the air a cryptiny value.
2) A nearby **observer** receives the cryptiny value, interprets the cryptiny as a public key and uses it to encrypt the observer's message (e.g. its location). The observer then forwards a (cryptographic) hash of the cryptiny value together with the observer's encrypted message to the cloud server over IP.
3) The **cloud server** maintains a mapping table which associates any received cryptiny-value hash with the corresponding beacon's owner. Given a cryptiny value and an observer's encrypted message the cloud server uses the cryptiny value as a key to the mapping table in order to determine the corresponding owner identifier. It then routes the hash of the cryptiny value together with the observer's encrypted message to that owner.
4) The **beacon's owner** receives the hash of the cryptiny value together with the observer's encrypted message. It uses the hash of the cryptiny value to find out the signal encrypted by its beacon and the private key required for decrypting the observer's message and decrypts the observer's message.

Note: Since using the cryptiny value as a public key is central to our protocol, we sometimes refer to the cryptiny value as the beacon's public key.

C. *Key Management*

The owner and its beacon exchange cryptographic keys in the initialization step during pairing. In addition, the owner assists the cloud server generate its mapping table by providing the cloud server with the relevant entries corresponding to that owner's beacon. In addition to its use by the cloud server to determine the respective owner given a beacon's cryptiny value as described above, the mapping table is used for certifying public keys as we now explain. Recall that the cryptiny value is interpreted by the observer as a public key and is used by the observer for encrypting its message. This immediately raises the issue of public key integrity. As discussed above, this cannot be achieved using PKI mechanisms. Instead, we use the following autonomous key certification: Before using a received cryptiny value as a public key, an observer verifies with the cloud server that the hash of the public key is in the server's mapping table. This guarantees that the public key is genuine (that it, it could have been generated by a beacon of one of the owners).

D. Security Model

1) The players:

- Beacon: clearly for the security of the beacon's signal and for the beacon's indistinguishability, the beacon is assumed to follow the protocol. However, we do not limit the beacon's behavior for the security of the observer's message.
- Observer: again, for the security of the observer's message, the observer is assumed to follow the protocol. However, we do not limit the observer's behavior for the security of the beacon's signal and for the beacon's indistinguishability.
- Cloud server: the cloud server is part of the service provider, and therefore has economic intensive to play faithfully. Hence it is assumed to be honest but curious. Nevertheless, we allow the adversary to use its knowledge of the cloud server's data to attack the protocol outside the cloud server (that is, without interfering with the cloud server protocol or modifying cloud server data). For example, such an adversary may generate manipulated data based on the cloud server's data, and send the manipulated data to the cloud server as if it were an observer. We refer to such an adversary as a **server-state reading** adversary.
- Owner: We require that all owners follow the protocol and that all owners' secrets keys are not compromised. The requirement that none of the owners' secrets keys are compromised stems from the fact that every owner in the protocol serves a role similar to a certification authority (CAs) where in our case, the hash of a cryptiny value provided by an owner to the cloud server serves as a certificate for that cryptiny value when it is interpreted as a public key by an observer.

2) The player's communication channels: As mentioned above, the owner and beacon communicate and exchange keys before the protocol starts. This communication is performed using cryptographic keys exchanged between the owner and beacon during the pairing stage which is assumed to take place in an isolated environment. We therefore assume that the beacon-owner channel is secure and authenticated. The observer-to-cloud-server channel and the cloud-server-to-owner channel are both over IP and are therefore assumed to be secure and authenticated as well (using a secure internet protocol such as TLS). However, we do not make any security assumptions with respect to the beacon-observer channel.

E. Security, Privacy, and Integrity guarantees

Security

- Beacon's signal CCA-security: The beacon's signal is CCA-secure (i.e., secure against chosen ciphertext attack). The adversary here is composed of the "normal" CCA adversary with the additional capabilities of a server-state-reading adversary.
- Observer's message CCA-security with backward and forward security: The observer's message is CCA-secure even if beacons' secrets are compromised and/or beacons do not follow their protocol. As before, the adversary here is composed of the standard CCA adversary with the additional capabilities of a server-state-reading adversary. Additionally we provide the adversary with all beacons' secrets and the ability to modify beacons' behaviour.

Privacy

- Beacon's indistinguishability: Beacons' broadcasts cannot be distinguished from random values. Here we assume that the adversary has no access to the cloud server's data.

Integrity

- Public key certification: Observers encrypt their messages only with certified public keys (as described in II-C).

F. What the Protocol does not Guarantee

In our system, the owner expects to receive an observer's message that was generated by an observer in the beacon's vicinity and where that observer follows the protocol. However since beacons broadcast over the air, a beacon's broadcasts can be forwarded to an observer in a different location (we refer to such an attack as a replay attack). Additionally, observers may deviate from the protocol and send fake messages. Such attacks can be handled using incentives and anti-abuse layers outside the cryptographic protocol.

For example, to deal with replay attacks, the owner can decide to accept an observer's message only if the same message is received from several observers. To prevent an observer which is a smartphone from faking its message, the observer's operating system may be put under the control of a MANET (Mobile Ad hoc NETwork) system which is outside of the user's control. The system should be designed in a way that breaking into the physical security of the phone will be detected by the OS provider as part of anti-abuse measures. Clearly these mechanisms (and others) can be defeated by a determined adversary. However, given the anti abuse layers, the cost of such attacks would in general be higher than their gains.

III. Formal Cryptographic Model Of the IoT-E2EE Problem

We formally model the IoT-E2EE problem and its correctness, security and privacy requirements which are derived from the discussion in the previous section.

Let the IoT-E2EE Problem

$$\mathsf{QURT} = (\mathsf{Init}, \{\mathsf{Bcn_i}\}_{i=1}^{w}, \mathsf{Obs}, \mathsf{Svr}, \{\mathsf{Own_i}\}_{i=1}^{w})$$

be a set of probabilistic polynomial-time algorithms where w is an input to the Init function. Specifically,

- The `initialization algorithm` Init takes as input a security parameter 1^n and 1^w, and outputs cryptographic and other parameters: For each $i \in [1, w]$ it generates cryptographic parameters $\mathcal{K}_i, \mathcal{K}_i'$ and a table Tbl_i. In addition it generates a table $\mathcal{M}$ ($\mathcal{M}$ is the mapping table). For each $i \in [1, w]$, $\mathcal{K}_i$ is intended for beacon i, and from now on we denote Bcn_i by $\mathsf{Bcn}_{\mathcal{K}_i}$. Similarly, for $i \in [1, w]$, $\mathcal{K}_i'$ and Tbl_i are intended for owner i, and from now on we denote Owner_i by $\mathsf{Owner}_{\mathcal{K}_i'}$. Finally, $\mathcal{M}$ is intended for the cloud server. We assume for simplicity that the tables $\mathcal{M}$ and Tbl_i for any owner i, are initialized once and for all. In practice, these tables will be generated piecewise on the fly (see Section IV-D).
- The `beacon's algorithm` $\mathsf{Bcn}_{\mathcal{K}_i}$ for $i \in [1, w]$, takes as input a time t and a signal s, and outputs its broadcast $\mathsf{Bcn}_{\mathcal{K}_i}(t, s)$ (this is the cryptiny value).
- The `observer's algorithm` Obs takes as an input x (expected to be a beacon's output $\mathsf{Bcn}_{\mathcal{K}_i}(t, s)$ in the observer's vicinity) and a message m, and outputs $\mathsf{Obs}(x, m)$.
- The `cloud server algorithm` Svr takes as an input y (expected to be an observer's output $\mathsf{Obs}(x, m)$), and outputs $\mathsf{Svr}(y)$.
- The `owner's algorithm` $\mathsf{Own}_{\mathcal{K}_i'}$ for $i \in [1, w]$, takes as an input z (expected to be a server's output intended for owner i), and outputs $\mathsf{Own}_{\mathcal{K}_i'}(z)$.

Definition 1: [QURT correctness] It is required that for every $i \in [1, w]$, time t, and signal s,

$$(s, m) = \mathsf{Own}_{\mathcal{K}_i'}(\mathsf{Svr}(\mathsf{Obs}(\mathsf{Bcn}_{\mathcal{K}_i}(t, s), m))).$$

In particular, correctness requires that $\mathsf{Svr}(\mathsf{Obs}(\mathsf{Bcn}_{\mathcal{K}_i}(t, s), m))$ is intended for owner i.

In Definition 2 and Definition 3 below, although not explicitly mentioned, adversary $\mathcal{A}$ includes all necessary accesses to be able to act also as a server-state reading adversary. The reason for that is that the table $\mathcal{M}$ and other information in the cloud server do not change and the protocol of the cloud server and owner are fixed.

In Definition 2 below, we implicitly assume that the public key with which the adversary encrypts its message is authentic (that is, it could have been generated by one of the legitimate beacons). This assumption is justified by II-C.

Definition 2: [Observer's message CCA-secure with backward and forward security] Let $i \in [1, w]$ be a beacon, and let $\mathcal{A}$ be an adversary in the following game $\mathrm{Exp}^{\mathrm{CCA}}_{\mathrm{ObsMsg},\mathcal{A}}(i, n)$:

- Adversary $\mathcal{A}$ receives access to:
  - The beacon's keys $\mathcal{K}_j$ for each $j \in [1, w]$ (the compromised beacon's secrets for backward and forward security).
  - The cloud server's mapping table oracle $\mathcal{M}$.
  - The owner's oracle $\mathsf{Own}_{\mathcal{K}_j'}()$ for all $j \in [1, w]$ (serves as a decryption oracle for CCA).
  - The adversary doesn't need access to the observers' oracles (equivalent to encryption oracle) since having all beacons' keys it can imitate both beacons' and observers' behavior.
- Adversary $\mathcal{A}$ chooses a desired time t_c, a signal s, and two distinct messages $m_0 \neq m_1$. It then sends them to the challenger. The time t_c chosen by the adversary can be any time in the past, present or future to guarantee forward and backward security.
- The challenger chooses a random bit $b \in \{0, 1\}$ and returns
$$y = \mathsf{Obs}(\mathsf{Bcn}_{\mathcal{K}_i}(t_c, s), m_b)$$
to adversary $\mathcal{A}$. As we see in Section V-B, the value y in our protocol is actually a triple $y = (y_1, y_2, y_3)$ where y_1 is the beacon's public key, y_2 is the observer's ephemeral public key, and y_3 is the symmetric encryption.
- Adversary $\mathcal{A}$ continues to have access to all oracles and keys as above, but cannot query the i'th owner's oracle $\mathsf{Own}_{\mathcal{K}_i'}()$ with $\mathsf{Svr}(y_1^v, y_2^{(1/v)}, y_3)$ for any v. The requirement with respect to the first two parameters makes sense even though the adversary has some control on the value of the first parameter by its knowledge of the beacons' keys (recall that we are proving backward security). This is due to the fact that in the actual protocol, the second parameter is the observer's ephemeral public key which is random and is thus independent of the first parameter.
- Adversary $\mathcal{A}$ returns b' and wins if $b' = b$.

We say that QURT achieves observer's message CCA-security with backward and forward security if for any PPT adversary $\mathcal{A}$ there exists a negligible function $negl$ such that for any beacon $i \in [1, w]$

$$\Pr[\mathrm{Exp}^{\mathrm{CCA}}_{\mathrm{ObsMsg},\mathcal{A}}(i, n) = 1] \leq \frac{1}{2} + negl(n).$$

Definition 3: [Beacon's signal CCA-secure] Let $i \in [1, w]$ be a beacon, and let $\mathcal{A}$ be an adversary in the following game $\mathrm{Exp}^{\mathrm{CCA}}_{\mathrm{BcnSgn},\mathcal{A}}(i, n)$:

- Adversary $\mathcal{A}$ receives access to:
  - The cloud server's mapping table oracle $\mathcal{M}$.
  - Beacons' oracles $\mathsf{Bcn}_{\mathcal{K}_j}()$ for all $j \in [1, w]$ (serve as encryption oracle in CCA).
  - Owners' oracles $\mathsf{Own}_{\mathcal{K}_j'}()$ for all $j \in [1, w]$ (serve as decryption oracle in CCA).
- Adversary $\mathcal{A}$ chooses two signals s_0, s_1 and t_c for which $\lfloor t_c/T \rfloor \neq \lfloor t/T \rfloor$ for all time t in the queries $\mathsf{Bcn}_{\mathcal{K}_i}()$ and $\mathsf{Own}_{\mathcal{K}_i'}()$ adversary $\mathcal{A}$ already made. It then sends t_c, s_0, s_1 to the challenger.
- The challenger chooses a random bit $b \in \{0, 1\}$ and returns $x = \mathsf{Bcn}_{\mathcal{K}_i}(t_c, s_b)$ to adversary $\mathcal{A}$.
- Adversary $\mathcal{A}$ continues to have access to its three oracles as before, where: (1) the parameter t for the i'th

beacon's oracle must obey $\lfloor t/T \rfloor \neq \lfloor t_c/T \rfloor$; and (2) the i'th owner's oracle $\mathsf{Own}_{\mathcal{K}'_i}()$ cannot be queried with $\mathsf{Svr}(\mathsf{Obs}(x, m))$ for any m (that is, we do not allow to decrypt the challenge x).

- Adversary $\mathcal{A}$ returns b' and wins if $b' = b$.

We say that QURT achieves beacon's signal CCA-security if for any PPT adversary $\mathcal{A}$ there exists a negligible function $negl$ such that for any beacon $i \in [1, w]$

$$\Pr[\mathrm{Exp}^{\mathrm{CCA}}_{\mathrm{BcnSgn},\mathcal{A}}(i, n) = 1] \leq \frac{1}{2} + negl(n).$$

Definition 4: [Beacon's Indistinguishability] Let $i \in [1, w]$ be a beacon, and let $\mathcal{A}$ be an adversary as follows:

- Adversary $\mathcal{A}$ receives access to oracle $O()$ which is either $\mathsf{Bcn}_{\mathcal{K}_i}()$ or a random oracle $\mathsf{GRand}()$ which returns a random value in the range of $\mathsf{Bcn}_{\mathcal{K}_i}$. [1]
- In addition, adversary $\mathcal{A}$ receives access to all beacons' oracles but the i'th beacon. Namely, it receives $\mathsf{Bcn}_{\mathcal{K}_j}()$ for all $j \in [1, w]$ s.t. $j \neq i$.

Let $\mathcal{A}^{\mathsf{Bcn}}$ be an adversary as above where $O = \mathsf{Bcn}_{\mathcal{K}_i}$ and let $\mathcal{A}^{\mathsf{Rnd}}$ be an adversary as above where $O = \mathsf{GRand}$.

We say that QURT achieves beacon's indistinguishability if for any PPT adversary as above there exists a negligible function $negl$ such that for any beacon $i \in [1, w]$

$$\left| \Pr[\mathcal{A}^{\mathsf{Bcn}}(i, 1^n) = 1] - \Pr[\mathcal{A}^{\mathsf{Rnd}}(i, 1^n) = 1] \right| \leq negl(n).$$

IV. The Cryptiny Value and its Usage in our Protocol

Below, we delineate the methodology for generating the cryptiny value.

A. Our Starting Point

Our starting point for the beacon's broadcast is a pseudorandom version of the Diffie-Hellman (DH) key-exchange where the beacon's broadcast is a single point on an elliptic curve. This point represents an ephemeral DH public key for a near-by observer to encrypt its message. In order to synchronise the pseudorandom broadcast between the owner and its beacon, we use time as a parameter in the pseudorandomness computation. In order to save beacon's battery, the beacon changes its broadcast every fixed time period of T seconds (e.g., $T = 1024$). Thus, the broadcast of beacon i at time t is

$$g^{\mathsf{PRF}_{k_i}(\lfloor t/T \rfloor)}$$

where PRF is a pseudo-random function, g is a base-point generator in the elliptic curve group, and k_i is a shared key between beacon i and owner i (these parameters are defined in detail in section V).

Below, we incorporate additional security and privacy functionalities, while maintaining the size of the beacon's broadcast and preserving its pseudorandomness.

[1]The range of GRand and $\mathsf{Bcn}_{\mathcal{K}_i}$ will be a group. The "G" in "GRand" stands for that.

B. Adding Beacon's Signal Encryption

To add an encryption of the beacon's signal without extending the beacon's broadcast, we "fold" the encryption of the signal s into the existing beacon's broadcast by adding the signal s as another parameter of the PRF. Hence, so far, the broadcast of beacon i at time t is:

$$g^{\mathsf{PRF}_{k_i}(\lfloor t/T \rfloor, s)}.$$

As we shall see in Section IV-D, decryption of a beacon's signal utilizes the fact that the number of different signals s and different time periods $\lfloor t/T \rfloor$ is relatively small.

C. Adding Backward (and Forward) Security for the Observer's Message

The problem with the beacon's broadcast described so far in Section IV-B is that an adversary which compromises the secret key k_i of beacon i, can reveal the DH private key $\mathsf{PRF}_{k_i}(\lfloor t/T \rfloor, s)$ associated with the beacon's broadcast $g^{\mathsf{PRF}_{k_i}(\lfloor t/T \rfloor, s)}$ at time t. Since a beacon's broadcast is used by an observer in order to encrypt its message, given beacon i's secret key k_i, the adversary can decrypt all past and future observers' messages that were encrypted using the beacon's broadcast.

Our solution adds backward security (and forward security) and is inspired by the security principle of "separation of duties," specifically as in key-insulated cryptography [9], [8] where a server "helps" an entity which may be susceptible to key extraction to periodically refresh its secret key. In our case the owner plays the role of the helping entity. Specifically, we provide owner i with a random value $r_i \in \mathbb{Z}_q$ and give g^{r_i} to beacon i. The beacon uses g^{r_i} as its base-point generator instead of g and applies the exact same operations as before (namely this modification is transparent to the beacon). Hence, the final broadcast of beacon i is

$$g^{r_i \cdot \mathsf{PRF}_{k_i}(\lfloor t/T \rfloor, s)}.$$

This is the cryptiny value.

This separation of duties between the owner and beacon neutralizes the risk of compromising the beacon's secret keys since the beacon is no longer the "holder of all secrets." Indeed, since r_i is known to the owner only, an adversary having the beacon's secret key k_i (and the non-secret value g^{r_i}) would be unable to compute the DH private key $r_i \cdot \mathsf{PRF}_{k_i}(\lfloor t/T \rfloor, s)$. Specifically, we will show that an observer's message encrypted with such a beacon's broadcast is backward and forward CCA-secure.

D. Beacon-to-Owner Mapping Table

To achieve real-time association of beacons' broadcasts with their owners, we are inspired by Eddystone-EID [7]. Specifically, we provide the cloud server with a table $\mathcal{M}$ associating the hash of any expected cryptiny value with the beacon's respective owner. We refer to this table as the "beacon-to-owner mapping table." (In practice the table $\mathcal{M}$ is generated piecewise as we mentioned in Section III and as will be discussed in more details towards the end of the current section). The mapping table stores hash of cryptiny values rather than the cryptiny values themselves to prevent a server-state reading adversary from maliciously generating fake observers' messages as if it were an observer next to a beacon.

An additional and important role of the mapping table is that it enables an autonomous public key certification in our protocol.

Let t_b be the system's starting time and let t_e be an upper bound on the system ending time so that $t_e - t_b$ is larger than the lifetime of the system (we can choose any value t_e such that $t_e - t_b$ is polynomial in the security parameter). In addition, let S be the set of all possible signals.

The mapping table is the following: Recall that each beacon i maintains a secret key k_i and a non-secret value g^{r_i}. Let H be a cryptographic hash function. Then, for each beacon i, for each distinct time $\lfloor t/T \rfloor$ where $t \in [t_b, t_e]$, and for each $s \in S$, the hash of the respective cryptiny value $g^{r_i \cdot \mathsf{PRF}_{k_i}(\lfloor t/T \rfloor, s)}$ is associated with owner i. Namely,

$$H(g^{r_i \cdot \mathsf{PRF}_{k_i}(\lfloor t/T \rfloor, s)}) \rightarrow i.$$

Since, generating a cryptiny value requires knowledge of the respective beacon's secret key k_i, the cloud server cannot generate the mapping table $\mathcal{M}$ by itself. Instead, each owner generates the expected cryptiny values of its beacon and sends them to the server. The server then unifies the received cryptiny values from all owners into the complete mapping table $\mathcal{M}$.

In addition to sending the cryptiny values to the cloud server, owner i also keeps a local table Tbl_i whose entries are hash of cryptiny values mapped to the corresponding time and signal, namely

$$H(g^{r_i \cdot \mathsf{PRF}_{k_i}(\lfloor t/T \rfloor, s)}) \rightarrow (\lfloor t/T \rfloor, s)$$

for all distinct time $\lfloor t/T \rfloor$ where $t \in [t_b, t_e]$ and for all $s \in S$. As we will see below, table Tbl_i is utilized by owner i to decrypt its beacon's signal.

Obviously, it is not practical to have a single mapping table for all (hash of) cryptiny values in the system's lifetime. In practice, we divide the time axis into consecutive non-overlapping time periods of, say, 24 hours, and provide the cloud server only with an mapping table corresponding to the current time period.[2] In Section VIII we discuss the typical sizes of the mapping table for real-world time-period parameters of different applications. Similarly, the table Tbl_i of owner i is not initialized once in advance but generated on the fly, in correspondence with the mapping table $\mathcal{M}$.

E. Encryption/Decryption of the Beacon's Signal and Observer's Message

1) Encryption: As we've seen above, the cryptiny value is the encryption of the beacon's signal. Encryption of the observer's message is performed by using the cryptiny value as a DH ephemeral public key.

[2]The server may maintain mapping tables for several consecutive time periods. One reason this may be needed is that the table entries are supplied to the server by owners, and owners may be offline some of the time.

2) Decryption: Decryption of the hash value of a cryptiny value $g^{r_i \cdot \mathsf{PRF}_{k_i}(\lfloor t/T \rfloor, s)}$ by owner i to get signal s, cannot be achieved by applying an inverse function, since this would require inverting the hash function, solving a discrete log problem and inverting the PRF. Instead, owner i uses its own local table Tbl_i to find the $(\lfloor t/T \rfloor, s)$ values corresponding to the hash of the cryptiny value.

To decrypt the observer's message, owner i uses $(\lfloor t/T \rfloor, s)$ to generate the exponent of the cryptiny value, namely the respective DH private key $r_i \cdot \mathsf{PRF}_{k_i}(\lfloor t/T \rfloor, s)$. [3]

According to the protocol, the owner accepts as valid any legitimate beacon's broadcast. We leave to the application the decision whether to accept or reject cryptiny values with old time t. For example, the decision could depend on (the difference between) the current time and the time corresponding with the cryptiny value.

V. The Cryptiny IoT-E2EE Protocol

A. Definitions and Preliminaries

In this protocol, we use a Diffie-Hellman-based encryption, in particular, DHIES [2]. Let us start with describing DHIES.

Definition 5: (Group Generator) Let $\mathsf{GroupGen}$ be a probabilistic polynomial-time (PPT) algorithm that, on a security parameter input 1^n, outputs a description of a cyclic group $\mathbb{G}$, its prime order q, and a generator $g \in \mathbb{G}$. Run $\mathsf{GroupGen}(1^n)$ to obtain the public parameters $(\mathbb{G}, q, g)$.

Definition 6: (DHIES [2]) Let $\mathsf{SYM} = (\mathcal{E}, \mathcal{D})$ be a private-key authenticated-encryption scheme. We run $\mathsf{GroupGen}(1^n)$ to obtain $(\mathbb{G}, q, g)$. Let KDF be a key derivation function $\mathsf{KDF} : \mathbb{G} \rightarrow \{0,1\}^n$. $\mathsf{DHIES} = (\overline{\mathcal{E}}, \overline{\mathcal{D}}, \overline{\mathcal{K}})$ is the following three-tuple of algorithms defining public-key encryption:

- $\overline{\mathcal{K}}$: Chooses a uniform $x \in \mathbb{Z}_q$, sets the public key $pk = g^x$, sets the private key $sk = x$, and returns (pk, sk).
- $\overline{\mathcal{E}}_{pk}(m)$: Chooses a uniform $z \in \mathbb{Z}_q$, computes g^z and sets $k = \mathsf{KDF}((pk)^z)$. Computes $c = \mathcal{E}_k(m)$ and outputs (g^z, c).
- $\overline{\mathcal{D}}_{sk}(\hat{c}, c)$: Returns $\perp$ if $\hat{c} \notin \mathbb{G}$. Else sets $k = \mathsf{KDF}((\hat{c})^{sk})$. Returns $m = \mathcal{D}_k(c)$ (note that m can be $\perp$).

Definition 7: (negligible) A function f from the natural numbers to the non-negative real numbers is negligible if for every positive polynomial p there exists an N_0 such that for all integers $n > N_0$ it holds that $f(n) < 1/p(n)$.
Throughout the paper, we chose a security parameter n which will determine the suitable space of keys and will be suitable for the desired security definition of all cryptographic functions.

Definition 8: (Oracle Diffie-Hellman Assumption ODH [2]) Run $\mathsf{GroupGen}(1^n)$ to obtain $(\mathbb{G}, q, g)$, let $\mathsf{KDF} : \mathbb{G} \rightarrow \{0,1\}^n$, and for $w \in \mathbb{Z}_q$ let $\mathsf{KDF}_w(X) := \mathsf{KDF}(X^w)$. The

[3]It is also possible to extend Tbl_i by storing in each entry the corresponding DH private key. This would save the DH private key generation at decryption time. Alternatively, it is possible to rid of Tbl_i altogether. In this case, owner i would compute for all $s \in S$ the hash of the corresponding cryptiny value of the current time t until a match is found.

ODH assumption is the following: For any PPT adversary $\mathcal{A}$, there exists a negligible function $negl$ such that

$$\Pr[u \xleftarrow{R} \mathbb{Z}_q; v \xleftarrow{R} \mathbb{Z}_q; \mathcal{A}^{\mathsf{KDF}_v(\cdot)}(g^u, g^v, \mathsf{KDF}(g^{uv})) = 1] -$$
$$\Pr[u \xleftarrow{R} \mathbb{Z}_q; v \xleftarrow{R} \mathbb{Z}_q; \mathcal{A}^{\mathsf{KDF}_v(\cdot)}(g^u, g^v, \{0,1\}^n) = 1]$$
$$\leq negl(n).$$

Definition 9: [$\text{Exp}^{\text{CCA}}_{\text{DHIES},\mathcal{A}}(n)$] Let $\mathcal{A}$ be an adversary in the following game $\text{Exp}^{\text{CCA}}_{\text{DHIES},\mathcal{A}}(n)$:

- The challenger randomly chooses a private key $sk = r \in \mathbb{Z}_q$ and sends the public key $pk = g^r$ to adversary $\mathcal{A}$.
- Adversary $\mathcal{A}$ has access to the decryption oracle $\overline{\mathcal{D}}_{sk}(\cdot,\cdot)$.
- Adversary $\mathcal{A}$ sends to the challenger two distinct messages $m_0 \neq m_1$.
- The challenger chooses a random bit $b \in \{0,1\}$ and sends $y = \overline{\mathcal{E}}_{pk}(m_b)$ to adversary $\mathcal{A}$.
- Adversary $\mathcal{A}$ continues to have access to the decryption oracle as before, but it cannot apply the decryption oracle on y.
- Adversary $\mathcal{A}$ returns b' and wins if $b' = b$.

Theorem 1: [2] If the Oracle Diffie-Hellman (ODH) assumption holds and the private-key authenticated-encryption scheme SYM used in DHIES is CCA-secure, then DHIES is CCA-secure. Namely, for any PPT adversary $\mathcal{A}$ there exists a negligible function $negl$ such that

$$\Pr[\text{Exp}^{\text{CCA}}_{\text{DHIES},\mathcal{A}} = 1] \leq \frac{1}{2} + negl(n).$$

Proof 1: In [2].

Definition 10: (Pseudorandom Function (PRF)) Let PRF be a keyed function $\mathsf{PRF} : \{0,1\}^n \times \{0,1\}^* \to \mathbb{Z}_q^*$ where the first parameter is the key and q is a parameter. For a key k, we denote $\mathsf{PRF}(k,t)$ by $\mathsf{PRF}_k(t)$. We say that PRF is a pseudorandom function if for all polynomial time distinguishers D there exists a negligible function $negl$ such that

$$\left|\Pr[D^{\mathsf{PRF}_k(\cdot)}(1^n) = 1] - \Pr[D^{\mathsf{Rand}(\cdot)}(1^n) = 1]\right| \leq negl(n)$$

where Rand is a random function of the same domain and range as PRF.

B. The Cryptiny IoT-E2EE Protocol In Detail

Let $\mathsf{DHIES} = (\overline{\mathcal{E}}, \overline{\mathcal{D}}, \overline{\mathcal{K}})$ be a public-key DHIES encryption scheme with group parameters $(\mathbb{G}, q, g)$. Let KDF be a key derivation function $\mathsf{KDF} : \mathbb{G} \to \{0,1\}^n$, let H be a cryptographic hash function $H : \mathbb{G} \to \{0,1\}^n$, and let PRF be a pseudorandom function $\mathsf{PRF} : \{0,1\}^n \times \{0,1\}^* \to \mathbb{Z}_q^*$.

In Section III we defined the interface of the five-tuple of algorithms

$$\mathsf{QURT} = (\mathsf{Init}, \{\mathsf{Bcn}_{\mathcal{K}_i}\}_{i=1}^{w}, \mathsf{Obs}, \mathsf{Svr}, \{\mathsf{Own}_{\mathcal{K}'_i}\}_{i=1}^{w}).$$

Below we complete the definition of these five algorithms by providing the algorithmic details.

Defining the Algorithmic Details of QURT*:*

- $\mathsf{Init}(1^n, 1^w)$:
  1) Beacon and Owner keys initialization: For each $i \in [1, w]$: chooses random $r_i \in \mathbb{Z}_q$, $k_i \in \{0,1\}^n$, and sets
  $$\mathcal{K}_i = (g^{r_i}, k_i), \mathcal{K}'_i = (r_i, k_i),$$
  where $\mathcal{K}_i$ is the i'th beacon keys and $\mathcal{K}'_i$ is the i'th owner keys.
  2) Owner's table initialization: For each $i \in [1, w]$, $t \in [t_b, t_e]$, and $s \in S$, generates the table
  $$\mathsf{Tbl}_i[H(g^{r_i \cdot \mathsf{PRF}_{k_i}(\lfloor t/T \rfloor, s)})] := (\lfloor t/T \rfloor, s),$$
  where Tbl_i is the table of the i'th owner.
  3) Cloud server mapping table initialization: For all $i \in [1, w]$, $t \in [t_b, t_e]$, and $s \in S$, generates the beacon-to-owner mapping table:
  $$\mathcal{M}[H(g^{r_i \cdot \mathsf{PRF}_{k_i}(\lfloor t/T \rfloor, s)})] := i.$$
- $\mathsf{Bcn}_{\mathcal{K}_i}(t, s)$: On input time t and signal s returns
$$\mathsf{Bcn}_{\mathcal{K}_i}(t, s) = g^{r_i \cdot \mathsf{PRF}_{k_i}(\lfloor t/T \rfloor, s)}.$$
- $\mathsf{Obs}(pk, m)$: On input pk and a message m, if $H(pk) \notin \mathcal{M}$ returns $\perp$, otherwise returns:
$$\mathsf{Obs}(pk, m) = (H(pk), \overline{\mathcal{E}}_{pk}(m)).$$
- $\mathsf{Svr}(h, \hat{c}, c)$: On input $(h, \hat{c}, c)$ where $(\hat{c}, c)$ is expected to be $\overline{\mathcal{E}}_{pk}(m)$ for some message m and public key pk, returns
$$\mathsf{Svr}(h, \hat{c}, c) = (\mathcal{M}[h], h, \hat{c}, c)$$
if $h \in \mathcal{M}$. Otherwise returns $\perp$. Note that in practice, the first parameter $\mathcal{M}[h]$ is only used by the server to find out the right owner and is not actually needed in the server output.
- $\mathsf{Own}_{\mathcal{K}'_i}(i, h, \hat{c}, c)$: If h not in Tbl_i then returns $\perp$. Otherwise, sets
$$(\lfloor t/T \rfloor, s) \leftarrow \mathsf{Tbl}_i[h] \text{ and } sk = r_i \cdot \mathsf{PRF}_{k_i}(\lfloor t/T \rfloor, s).$$
Returns
$$\mathsf{Own}_{\mathcal{K}'_i}(i, h, \hat{c}, c) = \overline{\mathcal{D}}_{sk}(\hat{c}, c)$$
which is m if authentication succeeds, and $\perp$ otherwise. We assumed above that the first parameter of $\mathsf{Own}_{\mathcal{K}'_i}$ is i since otherwise $\mathsf{Own}_{\mathcal{K}'_i}$ would not have been called.

VI. Security and Privacy Proofs

In this section we prove that our Cryptiny IoT-E2EE Protocol achieves all the security and privacy requirements.

A. New Primitive: Shifted-DHIES (SDHIES)

We define a generalized version of DHIES we call shifted-DHIES or in short SDHIES which will be useful in the security proof of the observer's message (Section VI-B). In this version the public key is a function of $\ell \in [1, q-1]$ where the party that encrypts can control this ℓ parameter. Namely, the public key is not fixed, and it may be changed at every call by the party that encrypts with it. Next, we formally define SDHIES and prove that SDHIES is CCA-secure under the ODH assumption by reduction to DHIES.

Definition 11 (SDHIES): Let $\mathsf{DHIES} = (\overline{\mathcal{E}}, \overline{\mathcal{D}}, \overline{\mathcal{K}})$ be the DHIES scheme with group parameters $(\mathbb{G}, q, g)$. Let $\mathsf{SDHIES} = (\overline{\mathcal{E}'}, \overline{\mathcal{D}'}, \overline{\mathcal{K}'})$ be the following three-tuple of algorithms defining a public-key scheme:

- $\overline{\mathcal{K}'}$: Chooses a uniform private key $sk \in \mathbb{Z}_q$, sets the corresponding public key $pk = g^{sk}$, and returns (pk, sk).
- $\overline{\mathcal{E}'}_{pk}(m, \ell)$: Calculates $(\hat{c}, c) = \overline{\mathcal{E}}_{pk^\ell}(m)$ and returns $(\hat{c}, c, \ell)$.
- $\overline{\mathcal{D}'}_{sk}(\hat{c}, c, \ell)$: Returns $\overline{\mathcal{D}}_{sk\cdot\ell}(\hat{c}, c)$ which is either $\perp$ or the corresponding message m.

Definition 12: [$\mathrm{Exp}^{\mathrm{CCA}}_{\mathrm{SDHIES},\mathcal{A}}(n)$] Let $\mathcal{A}$ be an adversary in the following game $\mathrm{Exp}^{\mathrm{CCA}}_{\mathrm{SDHIES},\mathcal{A}}(n)$:

- The challenger randomly chooses a private key $sk \in \mathbb{Z}_q$, sets the public key $pk = g^{sk}$, and gives pk to adversary $\mathcal{A}$.
- Adversary $\mathcal{A}$ has access to the decryption oracle $\overline{\mathcal{D}'}_{sk}(\cdot,\cdot,\cdot)$.
- Adversary $\mathcal{A}$ sends to the challenger two distinct messages $m_0 \neq m_1$ and $\ell \in [1, q-1]$.
- The challenger chooses a random bit $b \in \{0,1\}$ and sends $(\hat{y}, y, \ell) = \overline{\mathcal{E}'}_{pk}(m_b, \ell)$ to adversary $\mathcal{A}$.
- Adversary $\mathcal{A}$ continues to have access to the decryption oracle as before, but it cannot apply the decryption oracle on $(\hat{y}^{1/v}, y, \ell\cdot v)$ for any v. In other words, the adversary cannot apply a decryption oracle query (a, b, c) where both $a^c = \hat{y}^\ell$ and $b = y$ hold.
- Adversary $\mathcal{A}$ returns b' and wins if $b' = b$.

Theorem 2: If DHIES is CCA-secure, then SDHIES is CCA-secure. Namely, for any PPT adversary $\mathcal{A}$ there exists a negligible function $negl$ such that

$$\Pr[\mathrm{Exp}^{\mathrm{CCA}}_{\mathrm{SDHIES},\mathcal{A}} = 1] \leq \frac{1}{2} + negl(n).$$

Proof 2: To prove the above, let $\mathcal{A}$ be an adversary against the CCA-security of $\mathsf{SDHIES} = (\overline{\mathcal{E}'}, \overline{\mathcal{D}'}, \overline{\mathcal{K}'})$. We build an adversary $\mathcal{A}'$ against the CCA-security of $\mathsf{DHIES} = (\overline{\mathcal{E}}, \overline{\mathcal{D}}, \overline{\mathcal{K}})$:

- The DHIES challenger randomly chooses a private key $sk \in \mathbb{Z}_q$ and sends the public key $pk = g^{sk}$ to adversary $\mathcal{A}'$.
- Adversary $\mathcal{A}'$ sends pk to adversary $\mathcal{A}$ and runs adversary $\mathcal{A}$. To simulate an SDHIES decryption oracle call $\overline{\mathcal{D}'}_{sk}(\hat{c}, c, \ell)$, adversary $\mathcal{A}'$ returns $\overline{\mathcal{D}}_{sk}(\hat{c}^\ell, c)$. Indeed, both decryptions result with the same message since the shared key of $\overline{\mathcal{D}}_{sk}(\hat{c}^\ell, c)$ and of $\overline{\mathcal{D}'}_{sk}(\hat{c}, c, \ell)$ is the same:
$$(\hat{c}^\ell)^{sk} = \hat{c}^{sk\cdot\ell}.$$
- Adversary $\mathcal{A}$ sends its challenge m_0, m_1 and ℓ to adversary $\mathcal{A}'$. Adversary $\mathcal{A}'$ sends m_0, m_1 to its DHIES challenger.
- The DHIES challenger chooses a random bit $b \in \{0,1\}$ and returns
$$(\hat{y}, y) = \overline{\mathcal{E}}_{pk}(m_b).$$
- Adversary $\mathcal{A}'$ sends to adversary $\mathcal{A}$ the challenge
$$(\hat{y}^{1/\ell}, y, \ell).$$
The triple $(\hat{y}^{1/\ell}, y, \ell)$ is indeed a correct challenge for adversary $\mathcal{A}$ since $\hat{y}^{1/\ell}$ is a random group element (because $\hat{y}$ is), and since both challenges represent the same encrypted message, that is, they use the same shared key:
$$(\hat{y})^{sk} = (\hat{y}^{1/\ell})^{sk\cdot\ell}.$$
- Adversary $\mathcal{A}'$ continues to answer decryption queries as before. From definition $\mathrm{Exp}^{\mathrm{CCA}}_{\mathrm{SDHIES},\mathcal{A}}(n)$, and given that the challenge is $(\hat{y}^{1/\ell}, y, \ell)$, adversary $\mathcal{A}$ is forbidden from querying $(\hat{y}^{1/(\ell\cdot v)}, y, \ell\cdot v)$ for any v. This implies that adversary $\mathcal{A}'$ would not query its decryption DHIES oracle with its challenge
$$((\hat{y}^{1/(\ell\cdot v)})^{\ell\cdot v}, y) = (\hat{y}, y).$$
- Eventually, adversary $\mathcal{A}$ returns its guess b', and adversary $\mathcal{A}'$ returns this guess b' to its challenger.

Since the view of adversary $\mathcal{A}$ in $\mathrm{Exp}^{\mathrm{CCA}}_{\mathrm{SDHIES},\mathcal{A}}$ is identical to the view of adversary $\mathcal{A}$ when its experiment is simulated by adversary $\mathcal{A}'$, it holds that:

$$\Pr[\mathrm{Exp}^{\mathrm{CCA}}_{\mathrm{DHIES},\mathcal{A}'}(n) = 1] = \Pr[\mathrm{Exp}^{\mathrm{CCA}}_{\mathrm{SDHIES},\mathcal{A}}(n) = 1].$$

The assumed CCA-security of DHIES, thus implies Theorem 2.

B. Observer's Message CCA-Secure

Theorem 3: The observer's message is CCA-secure with backward and forward security according to Definition 2 where oracles are defined according to the protocol in Section V-B.

Proof 3: Below we prove a somewhat stronger claim by providing the adversary with more information. Specifically, instead of providing the adversary with a mapping table that maps hash of cryptiny values to the corresponding owners' IDs, we provide the adversary with a mapping table that maps the cryptiny value themselves to the corresponding owners' IDs.

Let $i \in [1, w]$. We aim at proving that for any PPT adversary $\mathcal{A}$ there exists a negligible function $negl$ such that

$$\Pr[\mathrm{Exp}^{\mathrm{CCA}}_{\mathrm{ObsMsg},\mathcal{A}}(i, n) = 1] \leq \frac{1}{2} + negl(n).$$

To this end, we build an adversary $\mathcal{A}'$ against the CCA-security of $\mathsf{SDHIES} = (\overline{\mathcal{E}'}, \overline{\mathcal{D}'}, \overline{\mathcal{K}'})$:

- The SDHIES challenger randomly chooses a private key $sk \in \mathbb{Z}_q$ and sends the public key $pk = g^{sk}$ to adversary $\mathcal{A}'$.

- Adversary $\mathcal{A}'$ runs adversary $\mathcal{A}$. To this end, for all $j \neq i$, adversary $\mathcal{A}'$ generates random r_j and k_j, and sends $\mathcal{K}_j = (g^{r_j}, k_j)$ to $\mathcal{A}$. For $j = i$, adversary $\mathcal{A}'$ generates random k_i, sets $g^{r_i} := pk$ (note that $r_i = sk$ is not known to the adversary $\mathcal{A}'$), and sends $\mathcal{K}_i = (g^{r_i}, k_i)$ to $\mathcal{A}$. To simulate the mapping table $\mathcal{M}$ and prepare for owner queries, adversary $\mathcal{A}'$ generates for all $j \in [1, w]$, for any $t \in [t_b, t_e]$, and for any $s \in S$:
$$\mathcal{M}[g^{r_j \cdot \mathsf{PRF}_{k_j}(\lfloor t/T \rfloor, s)}] := j,$$
$$\mathsf{Tbl}_j[g^{r_j \cdot \mathsf{PRF}_{k_j}(\lfloor t/T \rfloor, s)}] := (\lfloor t/T \rfloor, s).$$
To simulate $\mathsf{Own}_{\mathcal{K}'_j}(j, pk, \hat{c}, c)$ for any j, adversary $\mathcal{A}'$ does the following: first it extracts $(\lfloor t/T \rfloor, s) \leftarrow \mathsf{Tbl}_j[pk]$ and then calculates $\ell = \mathsf{PRF}_{k_j}(\lfloor t/T \rfloor, s)$. If $j = i$ adversary $\mathcal{A}'$ applies its SDHIES decryption oracle $\overline{\mathcal{D}'}_{sk}(\hat{c}, c, \ell)$. This is valid since $\overline{\mathcal{D}'}_{sk}(\hat{c}, c, \ell) = \overline{\mathcal{D}}_{sk}(\hat{c}^{\ell}, c)$. Otherwise, if $j \neq i$, it calculates $\overline{\mathcal{D}'}_{r_j}(\hat{c}, c, \ell)$.
- Adversary $\mathcal{A}$ sends t_c, s, m_0, m_1 to adversary $\mathcal{A}'$. Adversary $\mathcal{A}'$ calculates $\ell = \mathsf{PRF}_{k_i}(\lfloor t_c/T \rfloor, s)$ and sends m_0, m_1, ℓ to the SDHIES challenger.
- The SDHIES challenger chooses a random bit b and returns
$$(\hat{y}, y, \ell) = \overline{\mathcal{E}'}_{pk}(m_b, \ell)$$
to adversary $\mathcal{A}'$. Adversary $\mathcal{A}'$ then sends the challenge
$$(pk^{\ell}, \hat{y}, y)$$
to adversary $\mathcal{A}$.
- Adversary $\mathcal{A}$ continues to access the owners' oracles simulated by adversary $\mathcal{A}'$, but cannot query the i'th owner oracle with $\mathsf{Svr}(pk^{\ell \cdot v}, \hat{y}^{(1/v)}, y)$ for any v.
Notice that adversary $\mathcal{A}'$ can simulate any permitted owner oracle query of adversary $\mathcal{A}$ since the restriction on an owner oracle query of adversary $\mathcal{A}$ translates to the same restriction on a decryption oracle query of adversary $\mathcal{A}'$.
- Finally, adversary $\mathcal{A}$ returns a bit b'. Adversary $\mathcal{A}'$ returns this bit b'.

Since the view of adversary $\mathcal{A}$ in the experiment $\text{Exp}^{\text{CCA}}_{\text{ObsMsg},\mathcal{A}}$ is identical to the view of adversary $\mathcal{A}$ when its experiment is simulated by adversary $\mathcal{A}'$, it holds that:
$$\Pr[\text{Exp}^{\text{CCA}}_{\text{ObsMsg},\mathcal{A}}(i,n) = 1] = \Pr[\text{Exp}^{\text{CCA}}_{\text{SDHIES},\mathcal{A}'}(n) = 1].$$
From Theorem 2 we get
$$\Pr[\text{Exp}^{\text{CCA}}_{\text{ObsMsg},\mathcal{A}}(i,n) = 1] \leq \frac{1}{2} + negl(n).$$

C. Beacon's Signal CCA-Secure

Theorem 4: The beacon's signal is CCA-secure according to Definition 3 where oracles are defined according to the protocol in Section V-B.

Proof 4: As in Section VI-B, we prove a stronger claim by providing the adversary with a mapping table that maps the cryptiny value themselves to the corresponding owners' IDs. Let $i \in [1, w]$. We aim at proving that for any PPT adversary $\mathcal{A}$ there exists a negligible function $negl$ such that:
$$\Pr[\text{Exp}^{\text{CCA}}_{\text{BcnSgn},\mathcal{A}}(i,n) = 1] \leq \frac{1}{2} + negl(n). \quad (1)$$
To this end we define a new game $\text{Exp}^{\text{CCA}}_{\text{RandSgn},\mathcal{A}}(i,n)$, similar to $\text{Exp}^{\text{CCA}}_{\text{BcnSgn},\mathcal{A}}(i,n)$ with the following modifications: in the oracles $\mathcal{M}, \mathsf{Bcn}_{\mathcal{K}_i}, \mathsf{Own}_{\mathcal{K}'_i}$ the pseudorandom function $\mathsf{PRF}_{k_i}(\cdot,\cdot)$ is replaced with $\mathsf{Rand}(\cdot,\cdot)$, where $\mathsf{Rand}(\cdot,\cdot)$ is a random function returning a random value in $\mathbb{Z}_q^*$.

Next, we prove that for any PPT adversary $\mathcal{A}$, there exists a negligible function $negl$ such that
$$\Pr[\text{Exp}^{\text{CCA}}_{\text{BcnSgn},\mathcal{A}}(i,n) = 1] - \Pr[\text{Exp}^{\text{CCA}}_{\text{RandSgn},\mathcal{A}}(i,n) = 1] \leq negl(n). \quad (2)$$

To prove Equation (2), let $\mathcal{A}$ be an adversary in the experiment $\text{Exp}^{\text{CCA}}_{\text{BcnSgn},\mathcal{A}}$. We build an adversary $\mathcal{A}'$ attacking the pseudo randomness of PRF_{k_i}:

- Denote the oracle of $\mathcal{A}'$ by $O(\cdot,\cdot)$. That is, $O(\cdot,\cdot)$ may be either $\mathsf{PRF}_{k_i}(\cdot,\cdot)$ or the random oracle $\mathsf{Rand}(\cdot,\cdot)$.
- Adversary $\mathcal{A}'$ runs adversary $\mathcal{A}$. To this end, adversary $\mathcal{A}'$ chooses a random $r_j \in \mathbb{Z}_q$ for all $j \in [1, w]$, and $k_j \in \{0,1\}^n$ for all $j \in [1, w]$ s.t. $j \neq i$. Adversary $\mathcal{A}'$ simulates $\mathcal{M}$ and prepares for owner queries as follows. For all $t \in [t_b, t_e]$ and all $s \in S$ it generates
$$\mathcal{M}[g^{r_i \cdot O(\lfloor t/T \rfloor, s)}] := i$$
and
$$\mathsf{Tbl}_i[g^{r_i \cdot O(\lfloor t/T \rfloor, s)}] := (\lfloor t/T \rfloor, s).$$
For all j s.t. $j \neq i$, for all $t \in [t_b, t_e]$, and for all $s \in S$ it generates
$$\mathcal{M}[g^{r_j \cdot \mathsf{PRF}_{k_j}(\lfloor t/T \rfloor, s)}] := j$$
and
$$\mathsf{Tbl}_j[g^{r_j \cdot \mathsf{PRF}_{k_j}(\lfloor t/T \rfloor, s)}] := (\lfloor t/T \rfloor, s).$$
Adversary $\mathcal{A}'$ simulates $\mathsf{Bcn}_{\mathcal{K}_j}$ and $\mathsf{Own}_{\mathcal{K}'_j}$ for any $j \neq i$ trivially since it holds all the secret keys for these oracles. For $j = i$ adversary $\mathcal{A}'$ simulates $\mathsf{Bcn}_{\mathcal{K}_i}$ and $\mathsf{Own}_{\mathcal{K}'_i}$ using the key r_i and using $O(\cdot,\cdot)$ instead of $\mathsf{PRF}_{k_i}(\cdot,\cdot)$.
- Adversary $\mathcal{A}$ chooses s_0, s_1 and t_c for which $\lfloor t_c/T \rfloor \neq \lfloor t/T \rfloor$ for all time t in the queries $\mathsf{Bcn}_{\mathcal{K}_i}()$ and $\mathsf{Own}_{\mathcal{K}'_i}()$ adversary $\mathcal{A}$ already made. Adversary $\mathcal{A}$ then sends t_c, s_0, s_1 to adversary $\mathcal{A}'$. Adversary $\mathcal{A}'$ chooses a random bit b and returns the challenge
$$x = g^{r_i \cdot O(t_c, s_b)}.$$
- Adversary $\mathcal{A}$ continues to have access to all oracles as before but is not allowed to query the i'th beacon's oracle $\mathsf{Bcn}_{\mathcal{K}_i}()$ with time t such that $\lfloor t/T \rfloor = \lfloor t_c/T \rfloor$ and is not allowed to query the i'th owner's oracle $\mathsf{Own}_{\mathcal{K}'_i}()$ with $\mathsf{Svr}(\mathsf{Obs}(x, m))$ for any m.
- Finally, adversary $\mathcal{A}$ returns a bit b'. If $b' = b$ then adversary $\mathcal{A}'$ returns 1, otherwise, $\mathcal{A}'$ returns 0.

When $O(\cdot,\cdot) = \mathsf{PRF}_{k_i}(\cdot,\cdot)$, the view of $\mathcal{A}$ is as in $\text{Exp}^{\mathsf{CCA}}_{\text{BcnSgn},\mathcal{A}}$ and when $O(\cdot,\cdot) = \mathsf{Rand}(\cdot,\cdot)$, the view of $\mathcal{A}$ is as in $\text{Exp}^{\mathsf{CCA}}_{\text{RandSgn},\mathcal{A}}$. Therefore

$$\Pr[\mathcal{A}'^{\mathsf{PRF}_{k_i}(\cdot,\cdot)} = 1] = \Pr[\text{Exp}^{\mathsf{CCA}}_{\text{BcnSgn},\mathcal{A}}(i,n) = 1]$$

$$\Pr[\mathcal{A}'^{\mathsf{Rand}(\cdot,\cdot)} = 1] = \Pr[\text{Exp}^{\mathsf{CCA}}_{\text{RandSgn},\mathcal{A}}(i,n) = 1].$$

Since PRF_{k_i} is pseudorandom function, there exists a negligible function $negl$ such that

$$\big|\Pr[\text{Exp}^{\mathsf{CCA}}_{\text{BcnSgn},\mathcal{A}}(i,n) = 1] - \Pr[\text{Exp}^{\mathsf{CCA}}_{\text{RandSgn},\mathcal{A}}(i,n) = 1]\big| \le negl(n).$$

Namely Equation (2) holds. Since the value $\lfloor t_c/T \rfloor$ is only used once in the challenge, it follows that

$$\Pr[\text{Exp}^{\mathsf{CCA}}_{\text{RandSgn},\mathcal{A}}(i,n) = 1] = \frac{1}{2}.$$

Equation (1) thus follows from Equation (2).

D. Beacon's Indistinguishability

Theorem 5: The beacon achieves indistinguishability according to Definition 4 where oracles are defined according to the protocol in Section V-B.

Proof 5: Let $i \in [1,w]$. We prove that for any PPT adversary $\mathcal{A}$ there exists a negligible function $negl$ such that

$$\big|\Pr[\mathcal{A}^{\mathsf{Bcn}}(i,1^n) = 1] - \Pr[\mathcal{A}^{\mathsf{Rnd}}(i,1^n) = 1]\big| \le negl(n). \quad (3)$$

To this end, we build an adversary $\mathcal{A}'$ attacking the pseudo randomness of PRF_{k_i}:

- Denote the oracle of $\mathcal{A}'$ by $O(\cdot,\cdot)$. That is, $O(\cdot,\cdot)$ is either $\mathsf{PRF}_{k_i}(\cdot,\cdot)$ or the random oracle $\mathsf{Rand}(\cdot,\cdot)$.
- Adversary $\mathcal{A}'$ runs adversary $\mathcal{A}$. To this end, adversary $\mathcal{A}'$ randomly chooses k_j for all $j \in [1,w]$ s.t. $j \neq i$ and r_j for all $j \in [1,w]$ (including i). For a query $\mathsf{Bcn}_{\mathcal{K}_j}(\cdot,\cdot)$ adversary $\mathcal{A}'$ answers $g^{r_j \cdot \mathsf{PRF}_{k_j}(\lfloor t/T \rfloor, s)}$ if $j \neq i$ and $g^{r_i \cdot O(\lfloor t/T \rfloor, s)}$ if $j = i$.
- Finally adversary $\mathcal{A}'$ outputs the output of $\mathcal{A}$.

When $O(\cdot,\cdot) = \mathsf{PRF}_{k_i}(\cdot,\cdot)$, adversary $\mathcal{A}'$ runs $\mathcal{A}^{\mathsf{Bcn}}$ and when $O(\cdot,\cdot) = \mathsf{Rand}(\cdot,\cdot)$, adversary $\mathcal{A}'$ runs $\mathcal{A}$ with an oracle $g^{r_i \cdot \mathsf{Rand}(\cdot,\cdot)}$ and all oracles $\mathsf{Bcn}_{\mathcal{K}_j}(\cdot,\cdot)$ for $j \neq i$. Since $r_i \in \mathbb{Z}_q$ and $\mathsf{Rand}(\cdot,\cdot)$ returns a random element in $\mathbb{Z}_q^*$, the oracle $g^{r_i \cdot \mathsf{Rand}(\cdot,\cdot)}$ is in fact $\mathsf{GRand}(\cdot,\cdot)$, a random function from the domain of $\mathsf{Bcn}_{\mathcal{K}_j}(\cdot,\cdot)$ to its range $\mathbb{G}$. Thus when $O(\cdot,\cdot) = \mathsf{Rand}(\cdot,\cdot)$, adversary $\mathcal{A}'$ runs $\mathcal{A}^{\mathsf{Rnd}}$. To summarize

$$\Pr[\mathcal{A}'^{\mathsf{PRF}_{k_i}(\cdot,\cdot)}(1^n) = 1] = \Pr[\mathcal{A}^{\mathsf{Bcn}}(i,1^n) = 1]$$

$$\Pr[\mathcal{A}'^{\mathsf{Rand}(\cdot,\cdot)}(1^n) = 1] = \Pr[\mathcal{A}^{\mathsf{Rnd}}(i,1^n) = 1].$$

Since PRF_{k_i} is pseudorandom function, there exists a negligible function $negl$ such that

$$\big|\Pr[\mathcal{A}^{\mathsf{Bcn}}(i,1^n) = 1] - \Pr[\mathcal{A}^{\mathsf{Rnd}}(i,1^n) = 1]\big| \le negl(n).$$

This proves Equation (3).

VII. Correctness

Having shown security and privacy of the protocol, we will briefly claim correctness of our protocol.

Theorem 6: Our Cryptiny IoT-E2EE Protocol is correct according to Definition 1.

Proof 6: We need to show that

$$(s,m) = \mathsf{Own}_{\mathcal{K}_i'}(\mathsf{Svr}(\mathsf{Obs}(\mathsf{Bcn}_{\mathcal{K}_i}(t,s),m))).$$

Recall that $\mathsf{Bcn}_{\mathcal{K}_i}(t,s) = pk$ where $pk = g^{r_i \cdot \mathsf{PRF}_{k_i}(\lfloor t/T \rfloor, s)}$. Thus $\mathsf{Obs}(\mathsf{Bcn}_{\mathcal{K}_i}(t,s),m) = (H(pk), \overline{\mathcal{E}}_{pk}(m))$, and $\mathsf{Svr}(\mathsf{Obs}(H(pk), \overline{\mathcal{E}}_{pk}(m))) = (\mathcal{M}[H(pk)], H(pk), \overline{\mathcal{E}}_{pk}(m)) = (i, H(pk), \overline{\mathcal{E}}_{pk}(m))$. Therefore, owner i finds $(\lfloor t/T \rfloor, s)$ from $\mathsf{Tbl}_i[H(pk)]$, computes $sk = r_i \cdot \mathsf{PRF}_{k_i}(\lfloor t/T \rfloor, s)$ and uses it to decrypt $m = \overline{\mathcal{D}}_{sk}(\overline{\mathcal{E}}_{pk}(m))$.

VIII. Cost of the Protocol

We next consider the cost of each component in the Cryptiny IoT-E2EE Protocol (an analysis which has been presented with the design to the implementation team):

Beacon: Beacon i computes cryptiny values, where a cryptiny value is computed by a single group exponentiation (or more precisely a single elliptic curve multiplication) using the base g^{r_i}. This exponentiation can be performed very efficiently (following pre-computation) since the base is fixed (see [4]). The beacon broadcasts (the x-coordinate of) a single curve point. NIST recommends using 224-bit elliptic curves through year 2030 and 256-bit elliptic curves through and beyond year 2030.[4] Using these recommendations implies that the curve point broadcast by the beacon is of length 224 bit or 256 bit, respectively which is extremely small. The cryptiny value computation time is in milliseconds and its power consumption is negligible relative to the power consumption of the beacon's communication. In fact, the beacon can operate on a small battery for at least a full year.

Observer: The observer (which is a smartphone with much larger computation and power resources than those of the beacon) applies two group exponentiations per beacon in its vicinity: One exponentiation uses a fixed base (g) and the other uses a random base ($g^{r \cdot \mathsf{PRF}_k(t,s)}$) which the observer gets from a near by beacon. Again, the fixed-base exponentiation can benefit significantly from [4], but in any case computing two group exponentiations is reasonable and has negligible affect on the observer's battery (which is the primary concern with respect to observers).

Cloud server: The main complexity measure for the cloud server is the size of the mapping table. With 256-bit hash function and three different signal values, each entry in the table is composed of 128-bit for the hash of the cryptiny value (by using, e.g., only 128 out of 256 bits of the hash value) and a 64-bit owner ID. Therefore, the size of each table entry is 192 bits. Consider for example a period of 24 hours and a new beacon broadcast every 20 minutes, namely 72 daily

[4]https://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-57pt1r5.pdf

cryptiny values. Then, the number of entries per beacon per 24 hours is $72 \times 3 = 216$ (the '3' represents the three possible battery values). Therefore the mapping table per beacon is of size 216×192 bits=41.5 KB. We consider two (very) different application scenarios below. The first application is a small city's sensory data collection with 1000 beacons. The size of the mapping table in this case is 41.5 MB - very small indeed. The second application is asset tracking. Here we assume 10^8 beacons. The size of the mapping table in this case is therefore 4.15 TB. This is certainly reasonable for a company with 10^8 users (or to its cloud provider).

The mapping table is implemented as a hash table, and therefore the lookup operation is very fast, taking a few milliseconds.

Owner: For each beacon, the owner needs to generate its Tbl table entries periodically. The owner then keeps the table to enable decryption and sends only the table's cryptiny values to the server. Using the above parameters, each entry of Tbl requires 128 bits, 32 bits and 2 bits for the cryptiny value, time and signal, respectively. For 24 hours and cryptiny value change every 20 minutes, the table is of size 72×162 bits=12 KB. Computing Tbl entails 72×3=216 exponentiations for computing the required cryptiny values. Each group exponentiation is with a fixed base and can therefore be computed using the efficient algorithm of [4]. Due to the multiple exponentiations, the process of generating the Tbl table takes a few seconds but can nevertheless be done without affecting user experience by either (1) pre-computing the exponentiations during the phone's idle times; or (2) computing the entries in small batches during the day instead of all at once (and similarly sending the table's computed cryptiny values to the cloud server in batches during the day). Therefore, the Cryptiny IoT-E2EE Protocol is efficient and can be easily integrated in real-world applications (and, in fact, parts of it have already been adopted to and implemented within a concrete setting).

IX. Extensions

The versatility of our protocol leads to the following possibilities:

A. Observer's Complete Indistinguishability

Our protocol ensures that observers remain anonymous. This is assured by the fact that observers do not posses any unique string - all observers act in exactly the same way. The system must make sure that observers remain anonymous even given low-layer communication details such as the IP. This can be achieved by removing these low-layer communication details before they reach the cloud server or by using, for example, the TOR network. We note that anonymity of the observer is important since the observer willingness to help should not cost the observer its privacy.

B. Anonymous Two-Way Communication

Our Cryptiny IoT-E2EE Protocol can be extended to provide efficient anonymous two-way real-time communication between the owner and the observer. Such anonymous two-way communication enables new applications (see an example below).

As discussed above, having beacon-to-owner mapping enables anonymous efficient one-way communication. This communication is indeed anonymous - the owner and the observer cannot identify each other. Two-way communication can be achieved by having at the cloud server open sessions with both the owner and the observer and forwarding their respective messages to each other. Anonymity is thus preserved also in the case of two-way real-time communication (section IX-A discusses how to hide the identity of the observer also from the cloud server).

As described in the protocol, the observer encrypts information intended for the owner using a symmetric shared key based on Diffie-Hellman key exchange, where the owner's public key comes from the beacon. This symmetric key can be utilized in the rest of the two-way communication session to enable the owner and the observer to exchange authenticated and encrypted messages with each other. Notice that the beacon is used only for the initial key-exchange process; in the rest of the session, the observer and the owner communicate through the server without the beacon's help (so that the observer is free to leave the beacon's vicinity). Nevertheless, if required, the owner may force the observer to stay in the vicinity of its beacon for the duration of the session by requiring frequent key exchanges based on the beacon's broadcasts.

The observer's confidence in communicating with the correct party relies on its trust in the near-by beacon. Conversely, the owner's assurance stems from the belief that it is communicating with a party that is (or was) in close proximity to its beacon.

An interesting application that can leverage anonymous two-way communication is an anonymous auction for an item associated with the beacon. In this scenario, only observers located near the beacon are allowed to place bids. The auction protocol may involve multiple communication rounds between the anonymous bidder (observer) and the anonymous owner of the item (beacon's owner), and this can be achieved by two-way communication.

C. Further issues

We highlighted the new paradigm, the protocol, its setting, and its development, and the protocol's security as a stand alone one. We note that arguing UC-security and similar composition properties is an interesting question for future research.

X. Related Work

In this paper we suggest cryptiny as a new cryptographic paradigm. To motivate it and demonstrate its utility we considered an important problem in the IoT, the IoT-E2EE problem. Below we consider related work with respect to both the cryptiny paradigm and the IoT-E2EE problem.

A. The Cryptiny paradigm

There are works such as [6], [10] which re-use cryptographic keys in order to limit the number of keys. However, cryptiny is different, since its goal is to decrease the cryptographic footprint in terms of its length (in our case, the beacon's broadcast length), and not in terms of the number of keys employed.

B. The IoT-E2EE Problem and Protocol

We are not aware to any work with respect to the IoT-E2EE problem as defined in this paper so we consider below protocols for different versions of the IoT-E2EE problem. These protocols are only concerned with finding the location of assets and thus require less security guarantees.

Specifically, in Apple's FindMy [11], Samsung's SmartTag [12] and Tile [1] the beacon does not send a signal so obviously no beacon's signal security is provided. SmartTag and Tile also do not provide end-to-end security with respect to the observer's message (i.e., location). FindMy, on the other hand, provides end-to-end security with respect to the observer's message but **without** backward security. Since FindMy also does not support a beacon's signal, the beacon in FindMy is only required to broadcast one pseudorandom DH public key for the observer to encrypt its location without the need to "fold" into it additional functions. Therefore Apple's protocol does not need to use the cryptiny paradigm.

It is also worth noting that Apple's FindMy does not achieve efficient real-time communication nor public key certification as a result of not supporting beacon-to-owner mapping table. This lack of the beacon-to-owner association at the cloud server in FindMy provides strong anonymity to the owner with respect to the cloud server, but opens the door to two types of attacks. (i) attacks [5], [3] which use the FindMy network as a public database; and (ii) attacks where a beacon sends a weak key to an observer with the intention to reveal the observer's message (which is the location in the case of Apple's FindMy). Such an attack would succeed since the observer lacks the ability to validate the received public key. Furthermore, we note that Apple hasn't published neither a detailed description of the FindMy protocol nor cryptographic proofs of its security.

XI. Conclusions

This paper presents cryptiny, a novel cryptographic paradigm designed to compact communication transmitted over narrow channels by consolidating multiple cryptographic objectives into a single cryptographic value. Cryptiny proves invaluable in applications constrained by limited cryptographic data size. This includes systems utilizing BLE devices since broadcast size of such devices is severely limited.

We developed cryptiny as part of a the development of IoT-E2EE problem, a real-world problem in the IoT domain, which aims at protecting IoT beacons communicating with their owners via BLE-IP gateways. This demonstrates the necessity of cryptiny in certain bandwidth limited scenarios. Our Cryptiny IoT-E2EE Protocol enables a beacon to broadcast a short unfragmented broadcast message while achieving efficient real-time beacon-to-owner CCA-secure communication, efficient real-time observer-to-owner CCA-secure communication with backward and forward security, mutual privacy (i.e., indistinguishability) for the beacons and for the observers, and autonomous public key certification suitable to pairing based initiations.

Finally and looking forward, we believe that our IoT-E2EE Protocol can potentially become a standard/IETF-contribution and contribute to IETF discussions, given its unique and comprehensive security features, and its versatility and suitability for real-world IoT applications.

References

[1] Tile. https://www.theverge.com/2022/1/28/22906392/life360-tile-location-data-precise-aggregated-privacy.

[2] Abdalla, M., Bellare, M., and Rogaway, P. Dhaes: An encryption scheme based on the diffie-hellman problem. *IACR Cryptol. ePrint Arch. 1999* (1999), 7.

[3] Bellon, A., Yen, A., and Pannuto, P. Demo abstract: Tagalong: A free, wide-area data-muling service built on the airtag protocol. In *20th ACM Conference on Embedded Networked Sensor Systems* (2022).

[4] Brickell, E.F., G. D. M. K. W. D. Fast exponentiation with precomputation. In *Advances in Cryptology — EUROCRYPT' 92* (1993).

[5] Bräunlein, F. Send my: Arbitrary data transmission via apple's find my network, 2021. https://positive.security/blog/send-my (last visited 2022-12-13).

[6] Coron, J.-S., Joye, M., Naccache, D., and Paillier, P. Universal padding schemes for rsa. In *Advances in Cryptology—CRYPTO 2002: 22nd Annual International Cryptology Conference Santa Barbara, California, USA, August 18–22, 2002 Proceedings 22* (2002), Springer, pp. 226–241.

[7] David, L., Hassidim, A., Matias, Y., Yung, M., and Ziv, A. Eddystone-eid: Secure and private infrastructural protocol for ble beacons. *IEEE Transactions on Information Forensics and Security* (2022).

[8] Dodis, Y., Katz, J., Xu, S., and Yung, M. Strong key-insulated signature schemes. In *International Workshop on Public Key Cryptography* (2003), Springer, pp. 130–144.

[9] Dodis, Y., Luo, W., Xu, S., and Yung, M. Key-insulated symmetric key cryptography and mitigating attacks against cryptographic cloud software. In *Proceedings of the 7th ACM Symposium on Information, Computer and Communications Security* (2012), pp. 57–58.

[10] Haber, S., and Pinkas, B. Securely combining public-key cryptosystems. In *Proceedings of the 8th ACM Conference on Computer and Communications Security* (2001), pp. 215–224.

[11] Heinrich, A., Stute, M., Kornhuber, T., and Hollick, M. Who can find my devices? security and privacy of apple's crowd-sourced bluetooth location tracking system. *arXiv preprint arXiv:2103.02282* (2021).

[12] Yu, T., Henderson, J., Tiu, A., and Haines, T. Privacy analysis of samsung's crowd-sourced bluetooth location tracking system. *arXiv preprint arXiv:2210.14702* (2022).