Extended Diffie-Hellman Encryption for Secure and Efficient Real-Time Beacon Notifications

Liron David, Omer Berkman, Avinatan Hassidim, David Lazarov, Yossi Matias, and Moti Yung

Google Research

Abstract. Every computing paradigm involving communication requires new security protocols employing cryptography. For example, the Internet gave rise to TLS/SSL, and Mobile Computing gave rise to End to End Encryption protocols. In this paper, we address an emerging IoT paradigm involving beacons attached to things and security protocols associated with this new configuration.
Specifically, we address the "beacon notification problem," a critical IoT paradigm aims at providing secure and efficient real-time notifications from beacons to their owners. Since the beacon notification problem has not yet been formally defined, we begin by inspecting natural requirements based on the operational setting and establishing correctness, security, and privacy definitions through the use of cryptographic games.
To resolve this problem, we propose a novel cryptographic tool we call XDHIES, which is a considerable extension of available Diffie-Hellman encryption schemes. We then show a new notification protocol built upon XDHIES and we prove that this cryptographic protocol is secure and private and successfully meets all the above problem's requirements.

1 Introduction

The IoT paradigm we address in this paper is the following emerging prototypical information flow scenario: Assume Alice owns an item with no Internet connection and wishes to receive notifications about the item's status such as temperature, humidity, or battery level as well as notifications from nearby devices such as smartphones in the beacon's vicinity. To enable this, Alice attaches to her item a small broadcasting device, referred to as a beacon, which incorporates the required sensor. This beacon is paired with Alice's mobile phone, which enables the establishment of shared cryptographic keys for secure communication. During operation, the beacon broadcasts its ephemeral ID (EID) along with its status, which is encoded in a quantized format (e.g. low, med, high) and represented by a small number of bits.

A device with IP connection in proximity to the beacon (referred to as an "observer") "hears" the beacon's broadcasts and forwards the received information (EID and status) to a cloud server (i.e., a typical app cloud server), together with its own message (e.g. its geo location), see Figure 1. The cloud server, forwards a received pair of beacon's status and observer's message to the beacon's owner based on the EID in the beacon's broadcast.

We motivate the above scenario with the following two applications:

(1) Location-tracking A person traveling by plane wishes to track in real time the location and temperature of its suitcase. The suitcase location may be important in case the suitcase gets lost and its temperature may be vital if the suitcase contains items sensitive to temperature extremes. To this end, the owner attaches a beacon with a temperature sensor to the suitcase and an observer adds its geo-location before forwarding the resulting notification to the owner (through the cloud server). The beacon's status (namely, its temperature) and the observer's location should remain private, allowing only the owner to track its suitcase and find out its status.

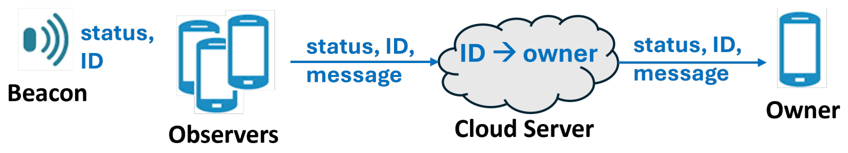

Fig. 1: The beacon notification configuration in IoT

(2) Proximity-based auction An anonymous donor has provided a valuable piece to a gallery for auction and desires (or is legally required) to limit real-time bids to individuals who can physically view and evaluate the piece. To facilitate this, the donor places a beacon next to the artwork, enabling observers in proximity to receive the beacon's broadcast, piggyback their bids and send the resulting notification to the owner (again through the cloud server). It is imperative that bidders, submitted bids and the auction winner remain confidential.

Here, the beacon's status may convey information about the temperature or humidity surrounding the piece to ensure that it is maintained within safe environmental conditions. This scenario may require a secure bi-directional connection to enable back-and-forth communication between bidders and the owner.

For example, we can assume that a winner can prove who it is (i.e., by presenting a long enough random message it can recite). Also, while our beacon notification protocol can be easily extended to accommodate bi-directional communication, in this paper we focus on the uni-direction bidding case only.

In the scenarios above, an adversary Eve aims at exposing beacons' status messages and observers' messages as well as compromising the owner's anonymity. To achieve secure and efficient notifications, we specifically require that a beacon notification protocol achieves the notions of security, privacy and integrity described below.

- Security: (1) A beacon's status message remains secret even if Eve has read access to cloud server data; (2) Any past or future message of an observer in the vicinity of a beacon remains secret even if Eve has read access to cloud server data and even if the beacon's keys are compromised (i.e., forward and backward secrecy).
- Privacy: Any beacon's broadcast must be pseudorandom and ephemeral to prevent tracking.
- Integrity: An observer can verify that a received beacon's broadcast is valid before piggybacking its own secure message and forwarding both to the cloud server.

1.1 Current State of the Art

Currently, only one protocol designed for non real-time secure beacon-location tracking, the Apple's FindMy protocol [12], exists. It is built upon a pseudorandom version of DHIES [2]. However, the FindMy protocol does not adhere to the same framework, as it neither provides a beacon's status nor supports real-time communication. Furthermore, it fails to meet our specified requirements for privacy, security, and integrity.

More specifically, the pseudorandom DHIES cryptosystem does not support:

- Beacon's status input: DHIES does not support a status input and therefore it does not provide security for the beacon's status. Appending the beacon's status independently to its broadcast would unwantedly increase the broadcast length. Since the beacon operates with low power consumption, it is crucial that its broadcasts remain brief, as extended broadcasts would deplete the battery. Additionally, broadcasting a single long message through multiple short transmissions poses a challenge, as it prevents observers which are briefly in proximity to the beacon from capturing an entire broadcast message. Therefore, enabling the beacon to securely broadcast its status without increasing its broadcast's length, that is without increasing the DHIES communication length, constitutes the first reason for extending the DHIES.

– Backward and forward security for observer's messages: The pseudorandom DHIES does not inherently support backward and forward security. Since the primary function of a beacon is to operate effectively when it is primarily distant from its owner, it becomes vulnerable to potential compromise. In such scenarios, adversaries may gain access to the beacon's secret keys and can therefore read any future/past message of an observer in this beacon's vicinity. Hence, achieving both backward and forward security is essential in beacon-based applications to ensure their overall security.

  Currently, backward security in DHIES is typically established through the process of re-pairing. However, this approach is inadequate for beacons, since they may remain separated from their owners for extended periods. If an adversary successfully exposes the beacon's secret keys, they can decrypt all future messages from observers in proximity to the beacon until the next encounter with the owner. This duration can extend to several days or even weeks. Forward security, in contrast, can be implemented in DHIES by applying a one-way function to the beacon's secrets at regular intervals, denoted as T. However, should an adversary uncover the beacon's secrets, they would be able to access all messages from observers within that time frame T and future messages beyond T.

  Thus, enabling the secure transmission of both future and past messages, independent of the duration T and the timing of the next encounter with the owner, constitutes the second reason for extending the DHIES.

Moreover, the existing routing in the FindMy protocol does not provide integrity, nor efficient real-time communication, and it is vulnerable to security attacks such as [6,4] (more details on these attacks in Section 10).

Consequently, a new cryptographic scheme and a new protocol are required to solve our notification problem in the setting above with the desired security and privacy requirements.

1.2 Contributions

Our central technical contribution is XDHIES, a novel and considerable extension of DHIES that integrates a status input and provides inherent backward and forward security for any past/future observer's message. Building on XDHIES (which may have other applications of independent interest), we propose a beacon notification protocol with a new routing scheme inspired by Eddystone-EID [9]. Unlike FindMy, our routing ensures integrity, enables efficient real-time communication, and mitigates security attacks [6,4]. Finally (and, again, unlike FindMy), we prove that our new beacon notification protocol meets the security and privacy requirements.

2 Threat Model and Requirements

In this section, we carefully and formally define the beacon notification problem, along with its associated correctness, security, privacy, and integrity requirements. These requirements are derived from the discussion above regarding the emerging new IoT setting.

2.1 Problem Definition

As described above, the beacon notification problem aims at forwarding both a status from a beacon and a message from an observer in this beacon's vicinity to the beacon's owner. The notification, therefore, includes both the beacon's status and the observer's message. Let t_b be the system's starting time and let t_e be the system ending time. We assume that the system's lifetime $t_e - t_b$ is polynomial in the security parameter n. In addition, let S be the set of all possible status messages.

Next we formally define the Beacon Notification Problem. Let

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

be a set of probabilistic polynomial-time algorithms. Specifically,

- The `initialization algorithm` Init takes as input a security parameter 1^n and a parameter 1^w indicating the number of owners in the system (for simplicity of presentation we assume that this number is known a priory), and outputs cryptographic parameters: (1) $\mathcal{K}_i$ intended for beacon i for each $i \in [1, w]$; (2) $\mathcal{K}'_i$ intended for owner i for each $i \in [1, w]$; and (3) $\mathcal{M}$ intended to the cloud server ($\mathcal{M}$ stands for a mapping table as will be described later).
- The `operation algorithm` which is a collection of events, each being an instance of one of the following atomic algorithms:
  - The `i'th beacon's algorithm` Bcn_i takes as input a time $t \in [t_s, t_e]$ and a status $s \in S$, and outputs its broadcast
  $$\mathsf{Bcn}_i(t, s).$$
  - An `observer` Obs takes as input a beacon's output x (which is expected to be $\mathsf{Bcn}_i(t, s)$ for some i, t and s) and a message m, and outputs
  $$\mathsf{Obs}(x, m).$$
  - The `cloud server algorithm` Svr takes as input an observer's output y (which is expected to be $\mathsf{Obs}(\mathsf{Bcn}_i(t, s), m)$ for some i, t, s and m) and outputs the index of the beacon that initiated y
  $$\mathsf{Srv}(y) := i.$$
  The cloud server then delivers y to the i'th owner.
  - The `i'th owner's algorithm` Own_i takes as input the cloud server's output y (which is expected to be $\mathsf{Obs}(\mathsf{Bcn}_i(t, s), m)$ for some t, s and m) and outputs
  $$\mathsf{Own}_i(y) := s, m.$$

2.2 Security

Beacon's status CCA-security Informally, we require that an adversary who can observe both a beacon's outputs and observers' outputs, and who has read-access to the cloud server data, would be unable to expose the beacon's status messages. Formally,

Definition 1. *(Beacon's status CCA-security) Let $i \in [1, w]$ be a beacon, let $\mathsf{Bcn} := \mathsf{Bcn}_i$, let $\mathsf{Own} := \mathsf{Own}_i$, and let $\mathcal{A}$ be an adversary in the following game $Exp^{CCA}_{BcnStat,\mathcal{A}}(n)$:*

- *Adversary $\mathcal{A}$ receives access to: (1) $\mathcal{M}$ (the cloud server's security parameter), (2) the beacon oracle $\mathsf{Bcn}()$, and (3) the owner oracle $\mathsf{Own}()$.*

- *Adversary $\mathcal{A}$ chooses two status messages s_0, s_1 and time t_c such that t_c is distinct from all times t in the queries to* $\mathsf{Bcn}()$ *and* $\mathsf{Own}()$ *that adversary $\mathcal{A}$ has 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 $x = \mathsf{Bcn}(t_c, s_b)$ to adversary $\mathcal{A}$.*
- *Adversary $\mathcal{A}$ continues to have access to $\mathcal{M}$ and the oracles as before, where: (1) the parameter t for any call to* $\mathsf{Bcn}()$ *must be distinct from t_c; and (2)* $\mathsf{Own}()$ *cannot be queried with $y = \mathsf{Obs}(x, m)$ 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 BcnNtf *achieves beacon's status CCA-security if for any PPT adversary $\mathcal{A}$ there exists a negligible function negl such that*

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

Observer's message CCA-security with backward and forward security Informally, we require that an adversary who can observe both a beacon's outputs and observers' outputs, has read-access to the cloud server data, and possesses knowledge of the beacon's secret keys, would be unable to expose any past/current/future messages from observers generated by this beacon's output. Note that providing an adversary with access to the beacon's secret keys is required to assure backward and forward security. Formally,

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

- *Adversary $\mathcal{A}$ receives access to: (1) $\mathcal{M}$ (the cloud server's security parameter), (2) the beacon's keys $\mathcal{K}$ (the compromised beacon's secrets for backward and forward security), and (3) the owner oracle* $\mathsf{Own}()$.
- *Adversary $\mathcal{A}$ chooses a desired time t_c, a status s, and two distinct messages $m_0 \neq m_1$. It then sends t_c, s, m_0, m_1 to the challenger.*
- *The challenger chooses a random bit $b \in \{0, 1\}$ and returns*

$$y = \mathsf{Obs}(\mathsf{Bcn}(t_c, s), m_b)$$

  to adversary $\mathcal{A}$.
- *Adversary $\mathcal{A}$ continues to have access to (1), (2) and (3) above, but cannot query* $\mathsf{Own}()$ *oracle with y.*
- *Adversary $\mathcal{A}$ returns b' and wins if $b' = b$.*

We say that BcnNtf *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*

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

2.3 Privacy

Informally, we require that an adversary observing the outputs of a beacon, would be unable to distinguish these outputs from random values. Formally,

Definition 3. *(Beacon's Indistinguishability) Let* $i \in [1, w]$ *be a beacon, let* $\mathsf{Bcn} := \mathsf{Bcn}_i$*, and let* $\mathcal{A}$ *be an adversary in the following game* $Exp_{BcnInd,\mathcal{A}}(n)$*:*

- *The challenger chooses a random bit* b*.*
- *If* $b = 0$*, the beacon's oracle* $\mathsf{Bcn}()$ *remains unchanged; otherwise if* $b = 1$ *then* $\mathsf{Bcn}()$ *is replaced with a random function* Rand *which returns random values in the range of* $\mathsf{Bcn}()$*.*
- *Adversary* $\mathcal{A}$ *guesses* b' *and wins if* $b' = b$*.*

We say that BcnNtf *achieves beacon's indistinguishability if for any PPT adversary as above there exists a negligible function negl such that*

$$\Pr[Exp_{BcnInd,\mathcal{A}}(n) = 1] \leq \frac{1}{2} + negl(n).$$

2.4 Unforgeability

Informally we require that the cloud server is unable to forge a valid beacon's broadcast despite the fact that it is equipped with the means of recognizing valid beacon broadcasts. Since beacon broadcast are pseudorandom this in turn means that the cloud server cannot generate valid observer messages.

Formally,

Definition 4. *(Beacon's Broadcast Unforgeability) Let* $i \in [1, w]$ *be a beacon, let* $\mathsf{Bcn} := \mathsf{Bcn}_i$*, and let* $\mathcal{A}$ *be an adversary in the following game* $Exp_{BcnFrg,\mathcal{A}}(n)$*:*

- *Adversary* $\mathcal{A}$ *receives read-access to* $\mathcal{M}$*.*
- *Adversary* $\mathcal{A}$ *wins if succeeds to generate* $\mathsf{Bcn}(t, s)$ *for some* t *and* s*.*

We say that BcnNtf *achieves beacon's broadcast unforgeability if for any PPT adversary* $\mathcal{A}$ *there exists a negligible function negl such that*

$$\Pr[Exp_{BcnFrg,\mathcal{A}}(n) = 1] \leq negl(n).$$

2.5 Integrity

Informally, we require that an observer is able to verify the reliability of a received beacon's broadcast before using it to encrypt its message. To this end, we provide the adversary with access to a beacon's outputs, observers' outputs, and read-access to the cloud server data, and require that it is unable to generate a value which is not a valid output of any beacon, but yet the observer accepts it. Formally,

Definition 5. *(Beacon's Integrity) Let* $\mathcal{A}$ *be an adversary in the following game* $Exp_{BcnInt,\mathcal{A}}(n)$*:*

- *Adversary* $\mathcal{A}$ *receives access to: (1)* $\mathcal{M}$*, (2) beacon's oracle* $\mathsf{Bcn}_i()$ *for any* $i \in [1, w]$*, and (3) owner's oracle* $\mathsf{Own}_i()$ *for any* $i \in [1, w]$*.*
- *Adversary* $\mathcal{A}$ *generates* v *and wins if (1) there does not exist* i, t, s *such that* $v = \mathsf{Bcn_i(t,s)}$*; and (2) there exists a message* m *for which* $\mathsf{Obs}(v, m) \neq \perp$*.*

We say that BcnNtf *achieves beacon's integrity if for any PPT adversary* $\mathcal{A}$ *there exists a negligible function negl such that*

$$\Pr[Exp_{BcnInt,\mathcal{A}}(n) = 1] \leq negl(n).$$

2.6 Correctness

Definition 6. *(*BcnNtf *correctness) It is required that for every $i \in [1, w]$, time $t \in [t_s, t_e]$, status $s \in S$, and message m*

$$(s, m) = \mathsf{Own}_{\mathsf{Svr}(\mathsf{Obs}(\mathsf{Bcn}_i(t,s),m))}(\mathsf{Obs}(\mathsf{Bcn}_i(t, s), m)).$$

3 Background: DHIES and Preliminaries

Our protocol extends the Diffie-Hellman public encryption scheme DHIES [2]. In this section we describe DHIES and its security and then describe the pseudo-random version of DHIES, we call PR-DHIES.

Definition 7. *(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 8. *(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* $\mathsf{KDF} : \mathbb{G} \to \{0, 1\}^n$. $\mathsf{DHIES} = (\overline{\mathcal{K}}, \overline{\mathcal{E}}, \overline{\mathcal{D}})$ *is the following three-tuple of algorithms defining public-key encryption:*

- $\overline{\mathcal{K}}$*: Chooses a uniform $x \in \mathbb{Z}_q$, sets the private key $sk = x$ intended for the decryption function $\overline{\mathcal{D}}_{sk}$, and outputs the public key $pk = g^{sk}$.*
- $\overline{\mathcal{E}}_{pk}(m)$*: 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 $k = \mathsf{KDF}((\hat{c})^{sk})$. If authentication succeeds it returns $m = \mathcal{D}_k(c)$, else returns $\perp$.*

Definition 9. *(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 10. *(Oracle Diffie-Hellman Assumption ODH [2]) Run* $\mathsf{GroupGen}(1^n)$ *to obtain $(\mathbb{G}, q, g)$, let* $\mathsf{KDF} : \mathbb{G} \to \{0, 1\}^n$*, and for $w \in \mathbb{Z}_q$ let* $\mathsf{KDF}_w(X) := \mathsf{KDF}(X^w)$*. The 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 11. *($Exp^{CCA}_{ASYM,\mathcal{A}}(n)$) Let $\mathcal{A}$ be an adversary in the following game $Exp^{CCA}_{ASYM,\mathcal{A}}(n)$:*

- *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 $b' = b$.*

We say that ASYM achieves CCA-security if for any PPT adversary $\mathcal{A}$ there exists a negligible function negl such that

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

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.*

Proof. In [2].

Definition 12. *(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.

3.1 PR-DHIES: Pseudorandom DHIES

In the DHIES protocol a random private key sk is utilized. However, this approach is not sufficient in our context: In the beacon notification problem, privacy concerns dictate that a beacon's broadcasts look to an eavesdropper independent from one another. Since a beacon's public key pk is used to encrypt messages to the beacon's owner, the beacon cannot independently select a random sk and broadcasts the corresponding pk. Instead, as is the case in the FindMy protocol, a pseudo-random version of DHIES is utilized in the beacon scenario. In this version, the beacon and its owner share a (symmetric) secret key x, and sk is generated based on a pseudo-random function of a shared nonce (in our case the current time t) keyed with x. This ensures that a beacon and its owner can independently generate the same private key sk.

This pseudo-random DHIES, which we denote "PR-DHIES", is defined as follows (see Figure 2):

Definition 13. *(PR-DHIES) 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* $\mathsf{KDF} : \mathbb{G} \to \{0,1\}^n$. $\mathsf{DHIES} = (\overline{\mathcal{I}}, \overline{\mathcal{K}}, \overline{\mathcal{E}}, \overline{\mathcal{D}})$ *is the following four-tuple of algorithms defining public-key encryption:*

- $\overline{\mathcal{I}}(n)$*: Chooses a uniform $x \in \{0,1\}^n$ intended for both the key-generation function $\overline{\mathcal{K}}_x$ and the decryption function $\overline{\mathcal{D}}_x$.*
- $\overline{\mathcal{K}}_x(t)$*: Sets the private-key $sk_t = \mathsf{PRF}_x(t)$ and outputs the corresponding public-key $pk_t = g^{sk_t}$.*
- $\overline{\mathcal{E}}_{pk_t}(m)$*: Chooses a uniform $z \in \mathbb{Z}_q$, computes g^z and sets $k = \mathsf{KDF}((pk_t)^z)$. Computes $c = \mathcal{E}_k(m)$ and outputs (g^z, c).*
- $\overline{\mathcal{D}}_x(t, (\hat{c}, c))$*: Returns $\perp$ if $\hat{c} \notin \mathbb{G}$. Else sets $sk_t = \mathsf{PRF}_x(t)$ and $k = \mathsf{KDF}((\hat{c})^{sk_t})$. Returns $m = \mathcal{D}_k(c)$ if authentication succeeds, and $\perp$ otherwise.*

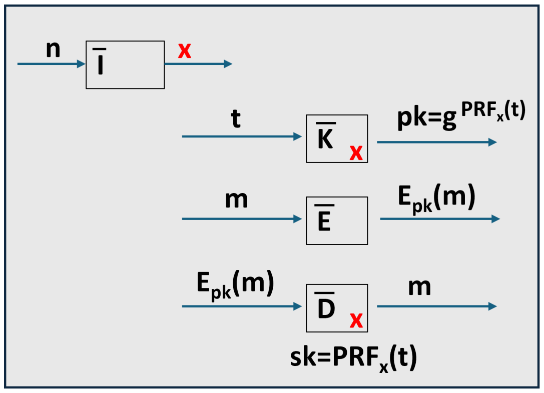

Fig. 2: PR-DHIES

Notice that the framework of public key encryption implemented by PR-DHIES is of the form $(\overline{\mathcal{I}}, \overline{\mathcal{K}}, \overline{\mathcal{E}}, \overline{\mathcal{D}})$ rather than the normal $(\overline{\mathcal{K}}, \overline{\mathcal{E}}, \overline{\mathcal{D}})$ framework. Specifically, this framework could be defined as follows:

Definition 14. *(Pseudorandom Public-Key Encryption) The framework consists of the following four probabilistic polynomial-time algorithms* $(\overline{\mathcal{I}}, \overline{\mathcal{K}}, \overline{\mathcal{E}}, \overline{\mathcal{D}})$*:*

- $\overline{\mathcal{I}}(n)$*: the initialization function* $\overline{\mathcal{I}}$ *takes a security parameter* n *and generates a secret key* $x \in \{0,1\}^n$*.*
- $\overline{\mathcal{K}}_x(t)$*: the key-generation function* $\overline{\mathcal{K}}_x$ *takes a time parameter* t *and returns a pseudorandom public key* pk_t*.*
- $\overline{\mathcal{E}}_{pk_t}(m)$*: the encryption function* $\overline{\mathcal{E}}_{pk_t}$ *takes a message* m *and returns its encryption.*
- $\overline{\mathcal{D}}_x(t, c)$*: the decryption function* $\overline{\mathcal{D}}_x$ *takes an encryption* c*, and decrypts it to get the respective* m*.*

This pseudorandom public-key encryption framework fails to meet two crucial requirements for beacons:

1. While such a framework allows a beacon to generate and broadcast a pseudorandom public key, it does not support additional inputs. This would force the beacon to broadcast its beacon status (in encrypted form) alongside the public key thus increasing the broadcast length. This is highly undesirable: The beacon is a small battery-powered device which may be in the field far from its owner for long time durations so that frequent battery replacement is not practical. 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. Hence the beacon's broadcast should be short and unfragmented.
2. The framework does not support backward and forward security. That is, if an adversary reveals the beacon's secret x, it can calculate the corresponding DH private-keys sk_t for all past and future values of t and can therefore decrypt any past/future messages. Backward and forward

security is a critical concern in beacon scenarios, as the beacon may be vulnerable to compromise when located far from its owner. Therefore, ensuring both forward and backward security is essential to protecting the confidentiality of observers' messages. Existing solutions for forward security apply a one-way-function to the beacon's secrets at regular interval, typically every T time units. However, if the beacon's secrets are compromised, the adversary can still decrypt all messages from observers within that T-time window. More crucially, for backward security, there is currently no solution other than re-pairing. Since beacons mainly operate at a distance from their owners, if the beacon's secrets are exposed, the adversary can read all messages from observers who were in the vicinity of this beacon until the next time this beacon meets its owner and re-establishes a secure connection with it. This can take days or even weeks.

We next define a new pseudorandom public-key encryption framework which extends the above $(\overline{\mathcal{I}}, \overline{\mathcal{K}}, \overline{\mathcal{E}}, \overline{\mathcal{D}})$ framework to address the missing requirements. We then establish the new framework's security criteria.

4 Extended Pseudorandom Public-Key Encryption Framework

We formally define the new framework and subsequently establish its security requirements.

4.1 Defining the Extended Framework

Definition 15. *(Extended Pseudorandom Public-Key Encryption) The new framework consists of the four probabilistic polynomial-time algorithms* $(\overline{\mathcal{I}}, \overline{\mathcal{K}}, \overline{\mathcal{E}}, \overline{\mathcal{D}})$*:*

- $\overline{\mathcal{I}}(n)$*: the initialization function* $\overline{\mathcal{I}}$ *takes a security parameter* n *and generates two sets of random keys* $\mathcal{K}$ *and* $\mathcal{K}'$*, where* $\mathcal{K}$ *is intended to the key-generation function and* $\mathcal{K}'$ *is intended to the decryption function.*
- $\overline{\mathcal{K}}_{\mathcal{K}}(t,s)$*: the key-generation function* $\overline{\mathcal{K}}_{\mathcal{K}}$ *takes a time parameter* t *and a small domain parameter* s*, and returns a pseudorandom public key* $pk_{t,s}$ *which incorporates the encryption of* s*.*
- $\overline{\mathcal{E}}_{pk_{t,s}}(m)$*: the encryption function* $\overline{\mathcal{E}}_{pk_{t,s}}$ *takes a message* m *and returns its encryption.*
- $\overline{\mathcal{D}}_{\mathcal{K}'}(c)$*: the decryption function* $\overline{\mathcal{D}}_{\mathcal{K}'}$ *takes an encryption* c*, and decrypts it to get the respective* s *and* m*.*

4.2 Defining the Framework's Security

Definition 16. *(*$Exp^{s\text{-}CCA}_{ASYM,\mathcal{A}}(n)$*) Let* $\mathsf{ASYM} = (\overline{\mathcal{I}}, \overline{\mathcal{K}}, \overline{\mathcal{E}}, \overline{\mathcal{D}})$ *be an extended pseudorandom public-key encryption scheme and let* $\mathcal{A}$ *be an adversary in the following game* $Exp^{s\text{-}CCA}_{ASYM,\mathcal{A}}(n)$*:*

- *The challenger randomly chooses* $\mathcal{K}, \mathcal{K}'$*.*
- *Adversary* $\mathcal{A}$ *has access to both the key-generation oracle* $\overline{\mathcal{K}}_{\mathcal{K}}(\cdot,\cdot)$ *and the decryption oracle* $\overline{\mathcal{D}}_{\mathcal{K}'}(\cdot)$*.*
- *Adversary* $\mathcal{A}$ *chooses two distinct messages* $s_0 \neq s_1$ *and* $t_c \in [t_s, t_e]$ *such that* t_c *is distinct from all values of* t *used in the queries above. It then sends* t_c, s_0, s_1 *to the challenger.*
- *The challenger chooses a random bit* $b \in \{0,1\}$ *and sends* $pk_{t_c,s_b} = \overline{\mathcal{K}}_{\mathcal{K}}(t_c, s_b)$ *to adversary* $\mathcal{A}$*.*
- *Adversary* $\mathcal{A}$ *continues to have access to both* $\overline{\mathcal{K}}_{\mathcal{K}}(\cdot,\cdot)$ *and* $\overline{\mathcal{D}}_{\mathcal{K}'}(\cdot)$ *as before, where: (1) the parameter* t *for any call must be distinct from* t_c*; and (2) the decryption oracle cannot be queried with* $y = \overline{\mathcal{E}}_{pk_{t_c,s_b}}(m)$ *for any* m *(that is, we do not allow to decrypt the challenge* pk_{t_c,s_b}*).*

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

We say that ASYM achieves s-CCA security if for any PPT adversary $\mathcal{A}$ there exists a negligible function negl such that

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

Definition 17. *($Exp^{m\text{-}CCA}_{ASYM,\mathcal{A}}(n)$) Let $\mathsf{ASYM} = (\overline{\mathcal{I}}, \overline{\mathcal{K}}, \overline{\mathcal{E}}, \overline{\mathcal{D}})$ be an extended pseudorandom public-key encryption scheme and let $\mathcal{A}$ be an adversary in the following game $Exp^{m\text{-}CCA}_{ASYM,\mathcal{A}}(n)$:*

– *The challenger randomly chooses $\mathcal{K}$ and $\mathcal{K}'$.*
– *Adversary $\mathcal{A}$ has access to $\mathcal{K}$ (the compromised secret keys for backward and forward security). It also has access to the decryption oracle $\overline{\mathcal{D}}_{\mathcal{K}'}(\cdot)$.*
– *Adversary $\mathcal{A}$ chooses t_c, a short domain message s_c, and two distinct messages $m_0 \neq m_1$ and sends t_c, s_c, m_0, m_1 to the challenger.*
– *The challenger calculates $pk_{t_c,s_c} = \overline{\mathcal{K}}_{\mathcal{K}}(t_c, s_c)$, chooses a random bit $b \in \{0, 1\}$, and returns*

$$y = \overline{\mathcal{E}}_{pk_{t_c,s_c}}(m_b)$$

to adversary $\mathcal{A}$.
– *Adversary $\mathcal{A}$ continues to have access to all oracles and keys as above, but cannot query the decryption oracle with y.*
– *Adversary $\mathcal{A}$ returns b' and wins if $b' = b$.*

We say that ASYM achieves m-CCA security with backward and forward security if for any PPT adversary $\mathcal{A}$ there exists a negligible function negl such that

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

Definition 18. *($Exp^{Ind}_{ASYM,\mathcal{A}}(n)$) Let $\mathsf{ASYM} = (\overline{\mathcal{I}}, \overline{\mathcal{K}}, \overline{\mathcal{E}}, \overline{\mathcal{D}})$ be an extended pseudorandom public-key encryption scheme and let $\mathcal{A}$ be an adversary in the following game $Exp^{Ind}_{ASYM,\mathcal{A}}(n)$:*

– *The challenger chooses a random bit b and $\mathcal{K}$.*
– *If $b = 0$, the key-generation function $\overline{\mathcal{K}}_{\mathcal{K}}$ remains unchanged; otherwise if $b = 1$ then $\overline{\mathcal{K}}_{\mathcal{K}}$ is replaced with a random function* Rand *which returns random values in the range of $\overline{\mathcal{K}}_{\mathcal{K}}$.*
– *Adversary $\mathcal{A}$ guesses b' and wins if $b' = b$.*

We say that ASYM achieves indistinguishability if for any PPT adversary as above there exists a negligible function negl such that

$$\Pr[Exp^{Ind}_{ASYM,\mathcal{A}}(n) = 1] = \frac{1}{2} + negl(n).$$

5 The XDHIES Scheme

We present XDHIES, an extended pseudorandom public key scheme. We explain below how XDHIES extends PR-DHIES to achieve the two crucial requirements discussed above.

5.1 Embedding Beacon's Status

To enable secure status transmission in the beacon's broadcast while preserving the length of a single DH public key, we incorporate the status as an additional input to the pseudo random function thus embedding (or "folding") the status within the public key. This approach allows the beacon to broadcast only the public key, which now includes the embedded status. Hence, the public key at time t becomes

$$pk = g^{\mathsf{PRF}_x(t,s)}$$

where x is the symmetric key shared between the beacon and its owner.

Obviously, decryption of the value pk to get status s cannot be achieved by applying an inverse function, since this would require solving a discrete log problem and inverting the PRF. Instead, since the number of different status messages s is relatively small (recall that the status s is composed of only a few bits), decryption of a beacon's status can be easily done in a brute-force manner or by maintaining a small table.

5.2 Providing Backward and Forward Security

The key idea enabling the beacon notification protocol to achieve backward and forward security is to prevent the beacon from holding critical secrets, rather than relying on one-way functions or re-pairing to update the beacon's secrets. In this way, even if an adversary compromises the beacon and reveals its secret key, it cannot decrypt any messages. Our solution is inspired by the security principle of "separation of duties," specifically as applied in key-insulated cryptography [11,10], where a server "helps" an entity susceptible to key extraction by periodically refreshing its secret key. In our case, the owner plays the role of the "helping" entity.

We designed XDHIES to incorporate such a secret separation. Specifically, XDHIES separates the secrets required for key-generation from those needed for decryption. We provide the beacon with only the information necessary for generating public keys and give the owner, who is the trusted entity, the information required for decryption, that is, the information necessary for generating the secret keys.

In particular, we provide the decryption function with a random value $r \in \mathbb{Z}_q$ and provide the key-generation function with g^r. The key-generation function uses g^r as its base-point generator instead of g and applies the exact same operations as before (i.e., this modification is transparent to the function). Consequently, r is known only to the owner who does not share it with the beacon. The beacon's public key at time t with status s is then:

$$pk = (g^r)^{\mathsf{PRF}_x(t,s)}.$$

This separation of responsibilities between the key-generation function (representing the beacon) and the decryption function (representing the owner) neutralizes the risk of compromising the beacon's secret keys since the beacon is no longer the "holder of all secrets." The DH private-key for decryption is then

$$sk = r \cdot \mathsf{PRF}_x(t,s).$$

As can be seen, even if an adversary reveals the beacon's secrets (x, g^r), it cannot reveal r, and is therefore unable to compute the DH private-key and decrypt past or future messages based on this DH-private key.

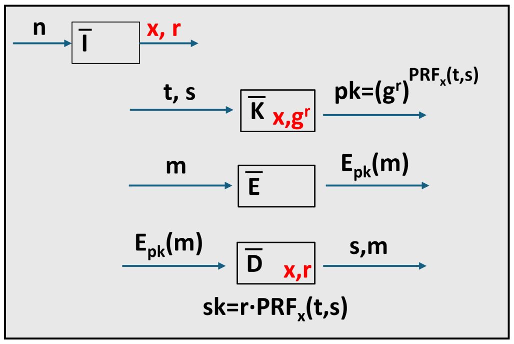

Fig. 3: XDHIES

5.3 XDHIES Definition

Definition 19. *(XDHIES) 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* $\mathsf{KDF} : \mathbb{G} \to \{0,1\}^n$*.* $\mathsf{XDHIES} = (\overline{\mathcal{I}}, \overline{\mathcal{K}}, \overline{\mathcal{E}}, \overline{\mathcal{D}})$ *is the following four-tuple of algorithms defining extended pseudorandom public-key encryption:*

- $\overline{\mathcal{I}}(n)$*: Chooses a uniform* $x \in \{0,1\}^n$ *and a random* $r \in \mathbb{Z}_q$*. Additionally generates table* Tbl *for the decryption function as follows: for any* $s \in S$ *and* $t \in [t_s, t_e]$ *sets* $pk_{t,s} = g^{r\cdot \mathsf{PRF}_x(t,s)}$ *and stores the entry* $(t,s) = \mathsf{Tbl}[pk_{t,s}]$*. Sets* $\mathcal{K} = (x, g^r)$ *which is intended for the key-generation function and sets* $\mathcal{K}' = (x, r, \mathsf{Tbl})$ *which is intended for the decryption function.*
- $\overline{\mathcal{K}}_{x,g^r}(t,s)$*: For a given pair of* (t,s) *sets the corresponding private key* $sk_{t,s} = \mathsf{PRF}_x(t,s)$ *and returns the corresponding public key* $pk_{t,s} = (g^r)^{sk_{t,s}}$*.*
- $\overline{\mathcal{E}}_{pk_{t,s}}(m)$*: Chooses a uniform* $z \in \mathbb{Z}_q$*, computes* g^z *and sets* $k = \mathsf{KDF}((pk_{t,s})^z)$*. Computes* $c = \mathcal{E}_k(m)$ *and outputs* (g^z, c)*.*
- $\overline{\mathcal{D}}_{x,r,\mathsf{Tbl}}(h, \hat{c}, c)$*: Returns* $\perp$ *if* $h \notin \mathsf{Tbl}$ *or if* $\hat{c} \notin \mathbb{G}$*. Otherwise, applies* $(t,s) = \mathsf{Tbl}[h]$*, sets* $sk_{t,s} = r \cdot \mathsf{PRF}_x(t,s)$ *and calculates* $k = \mathsf{KDF}((\hat{c})^{sk_{t,s}})$*. Returns* s *and* $m = \mathcal{D}_k(c)$ *if authentication succeeds.*

6 Security Proofs for XDHIES

Theorem 2. *XDHIES is s-CCA secure.*

Proof. To prove this, we define XDHIES-R, a variant of XDHIES where we replace every appearance of the pseudorandom function $\mathsf{PRF}_x(\cdot,\cdot)$ with $\mathsf{Rand}(\cdot,\cdot)$, where $\mathsf{Rand}(\cdot,\cdot)$ is a random function returning a random value in $\mathbb{Z}_q^*$.

Clearly, for any PPT adversary $\mathcal{A}$, there exists a negligible function *negl* such that

$$\Pr[\mathrm{Exp}^{\text{s-CCA}}_{\text{XDHIES},\mathcal{A}}(n) = 1] - \Pr[\mathrm{Exp}^{\text{s-CCA}}_{\text{XDHIES-R},\mathcal{A}}(n) = 1] \leq negl(n).$$

The challenge in the s-CCA game of XDHIES-R is $pk_{t_c,s_b} = g^{r\cdot\mathsf{Rand}(t_c,s_b)}$. Since the value t_c is only used to generate the challenge (and is not used in any of the oracle calls) and since the encryption is a random group element, it follows that

$$\Pr[\mathrm{Exp}^{\text{s-CCA}}_{\text{XDHIES-R},\mathcal{A}}(n) = 1] = \frac{1}{2}.$$

Therefore Theorem 7 follows.

Theorem 3. *XDHIES is m-CCA secure with backward and forward security.*

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

- The DHIES challenger randomly chooses a private key $r \in \mathbb{Z}_q$ and sends the public key $pk = g^r$ to adversary $\mathcal{A}'$.
- Adversary $\mathcal{A}'$ generates $x \in \{0,1\}^n$ and sends x, pk to adversary $\mathcal{A}$. Additionally, adversary $\mathcal{A}'$ generates Tbl with entries $(t,s) = \mathsf{Tbl}[pk^{\mathsf{PRF}_x(t,s)}]$ for any $s \in S$ and $t \in [t_s, t_e]$.
- Adversary $\mathcal{A}'$ runs adversary $\mathcal{A}$ and simulates an XDHIES decryption oracle call $\overline{\mathcal{D}}_{x,r,\mathsf{Tbl}}(h, \hat{c}, c)$ as follows: adversary $\mathcal{A}'$ returns $\perp$ if $h \notin \mathsf{Tbl}$ or $\hat{c} \notin \mathbb{G}$, and otherwise applies $(t,s) = \mathsf{Tbl}[h]$ and returns the pair $(s, \overline{\mathcal{D}'}_r(\hat{c}^{\mathsf{PRF}_x(t,s)}, c))$.
  Indeed, the shared symmetric decryption key in $\overline{\mathcal{D}'}_r(\hat{c}^{\mathsf{PRF}_x(t,s)}, c)$ is $(\hat{c}^{\mathsf{PRF}_x(t,s)})^r$ which is equal to $\hat{c}^{r \cdot \mathsf{PRF}_x(t,s)}$, the shared symmetric decryption key in $\overline{\mathcal{D}}_{x,r,\mathsf{Tbl}}(h, \hat{c}, c)$.
- Adversary $\mathcal{A}$ chooses t_c, a short domain message s_c, and two distinct messages $m_0 \neq m_1$ and sends t_c, s_c, m_0, m_1 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 sends to adversary $\mathcal{A}'$

$$(y_1, y_2) = \overline{\mathcal{E}'}_{g^r}(m_b).$$

- Adversary $\mathcal{A}'$ sends to adversary $\mathcal{A}$ the challenge

$$y = (pk^{\mathsf{PRF}_x(t_c,s_c)}, y_1^{1/\mathsf{PRF}_x(t_c,s_c)}, y_2).$$

  The pair $(y_1^{1/\mathsf{PRF}_x(t_c,s_c)}, y_2)$ is indeed a correct challenge for adversary $\mathcal{A}$ since $y_1^{1/\mathsf{PRF}_x(t_c,s_c)}$ is a random group element (because y_1 is), and since

$$(y_1)^r = (y_1^{1/\mathsf{PRF}_x(t_c,s_c)})^{r \cdot \mathsf{PRF}_x(t_c,s_c)}.$$

- Denote $(\overline{y_0}, \overline{y_1}, y_2) := y$. Adversary $\mathcal{A}$ continues to have access to the decryption oracle as above, but we forbid the adversary from querying the decryption oracle with $y_v = (\overline{y_0}^v, \overline{y_1}^{(1/v)}, y_2)$ for any v. While this requirement is stronger than not querying with y as required in the m-CCA definition (Definition 17), this requirement is reasonable since $\overline{y_1}$ is an ephemeral public key which is random and is thus independent of $\overline{y_0}$.
- 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 $\mathrm{Exp}^{\text{m-CCA}}_{\text{XDHIES},\mathcal{A}}$ is identical to the view of adversary $\mathcal{A}$ when its experiment is simulated by adversary $\mathcal{A}'$, it holds that:

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

The assumed CCA-security of DHIES, thus implies Theorem 3.

Theorem 4. *XDHIES achieves indistinguishability.*

Proof. Since $\overline{\mathcal{K}}_{x,g^r}(t,s) = g^{r \cdot \mathsf{PRF}_x(t,s)}$, and the adversary has access only to $\overline{\mathcal{K}}_{x,g^r}(\cdot,\cdot)$, the theorem is implied from the pseudorandomness of PRF_x.

7 The Beacon Notification Protocol

Let $\mathsf{XDHIES} = (\overline{\mathcal{I}}, \overline{\mathcal{K}}, \overline{\mathcal{E}}, \overline{\mathcal{D}})$ be the XDHIES encryption scheme described above with group parameters $(\mathbb{G}, q, g)$. Let H be a cryptographic hash function $H : \mathbb{G} \rightarrow \{0,1\}^n$, and let PRF be a pseudorandom function $\mathsf{PRF} : \{0,1\}^n \times \{0,1\}^* \rightarrow \mathbb{Z}_q^*$.

Our beacon notification protocol is built upon the XDHIES scheme. In Section 2.1 we define five-tuple of algorithms

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

which constitute the interface of the beacon notification protocol. Below we complete the definition of these five algorithms by providing the algorithmic details. We start with some general details and then delve into each algorithm.

First, a beacon broadcasts every second or two, but in practice it only changes its broadcast every fixed time period T (e.g. 15 minutes) in order to save battery.

Second, to achieve efficient real-time communication, enable integrity and mitigate security attacks such as [6,4], the design of the cloud server in our beacon notification protocol is inspired by Eddystone-EID [9]. In particular, the cloud server is provided with a mapping table $\mathcal{M}$ associating the beacons' broadcasts with the beacons' respective owners. That is, for each beacon i the mapping table $\mathcal{M}$ has an entry for every possible beacon's broadcast in $[t_s, t_e]$. We refer to the table $\mathcal{M}$ as the "beacon-to-owner mapping table." We note that in practice $\mathcal{M}$ is generated piecewise on the fly: the time axis is divided into consecutive non-overlapping time periods of, say, 24 hours, and at the start of each time period the cloud server is provided only with the part of the mapping table corresponding to the that time period.

Third, generating the beacons' broadcasts requires knowledge of the respective beacon's secret key, so that the cloud server cannot generate the mapping table $\mathcal{M}$ by itself. Instead, each owner generates the possible broadcast values of its beacon and sends them to the server. The server then unifies the received values from all owners into the complete mapping table $\mathcal{M}$.

Fourth, recall that a beacon's broadcast is the beacon's public key (embedded with the beacon's status). To prevent the cloud server from using these public keys to forge observers' messages, the cloud server mapping table $\mathcal{M}$ is keyed by a cryptographic hash of the beacons' public keys instead of the public keys themselves.

– $\mathsf{Init}(1^n, 1^w)$:
  1. Beacon and Owner keys initialization: For each $i \in [1, w]$: apply $\overline{\mathcal{I}}(n)$ to get random $r_i \in \mathbb{Z}_q$ and $x_i \in \{0,1\}^n$.
  2. Owner tables initialization: For any $i \in [1, w]$, generates Tbl_i as follows: For any $s \in s$ and $t \in [t_s, t_e]$

$$\mathsf{Tbl}_i[H(g^{r_i \cdot \mathsf{PRF}_{x_i}(\lfloor t/T \rfloor, s)})] := (\lfloor t/T \rfloor, s).$$

  3. Cloud server keys initialization: Generating $\mathcal{M}$ as follows: for all $i \in [1, w]$, $t \in [t_b, t_e]$, and $s \in S$:

$$\mathcal{M}[H(g^{r_i \cdot \mathsf{PRF}_{x_i}(\lfloor t/T \rfloor, s)})] := i.$$

4. Key distribution: Let
$$\mathcal{K}_i = (x_i, g^{r_i}), \mathcal{K}'_i = (x_i, r_i, \mathsf{Tbl}_i).$$
$\mathcal{K}_i$ is given to the i'th beacon, $\mathcal{K}'_i$ is given to the i'th owner and $\mathcal{M}$ is given to the cloud server. We note that each pair of $\mathcal{K}_i$ and $\mathcal{K}'_i$ is generated locally in private by the respective owner and $\mathcal{M}$ is combined by the cloud server from the pieces sent by all owners as described above.

– $\mathsf{Bcn}_i(t, s)$: On input time t and status s, applies $\overline{\mathcal{K}}_{x_i, g^{r_i}}$ to get $pk_{i,t,s}$, and returns
$$\mathsf{Bcn}_i(t, s) = pk_{i,t,s} = g^{r_i \cdot \mathsf{PRF}_{x_i}(\lfloor t/T \rfloor, s)}.$$

– $\mathsf{Obs}(pk, m)$: 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 h is expected to be the hash value of the i'th beacon's public-key $pk_{i,t,s}$ for some i, t and s and $(\hat{c}, c)$ is expected to be $\overline{\mathcal{E}}_{pk_{i,t,s}}(m)$ for some message m, if $h \notin \mathcal{M}$ it returns $\perp$; otherwise forwards $(h, (\hat{c}, c))$ to the corresponding owner which is
$$\mathsf{Svr}(h, (\hat{c}, c)) = \mathcal{M}[h] = i.$$

– $\mathsf{Own}_i(h, (\hat{c}, c))$: If $h \notin \mathsf{Tbl}_i$ returns $\perp$; otherwise, returns
$$\mathsf{Own}_i(h, (\hat{c}, c)) = \overline{\mathcal{D}}_{x_i, r_i, Tbl_i}(h, \hat{c}, c)$$
which is s, m if authentication succeeds, and $\perp$ otherwise.

8 Protocol Security and Privacy Proofs

In this section we prove that our beacon notification protocol achieves the security, privacy, unforgeability, integrity, and correctness requirements.

Theorem 5. *(Beacon's status CCA-security) The beacon notification protocol achieves beacon's status CCA-security according to Definition 1.*

Proof. Implied from the s-CCA security of XDHIES (Theorem 2).

Theorem 6. *(Observer's message CCA-security) The beacon notification protocol achieves observer's message CCA-security with backward and forward security according to Definition 2.*

Proof. Implied from the m-CCA security of XDHIES (Theorem 3).

Theorem 7. *(Beacon indistinguishability) The beacon notification protocol achieves indistinguishability according to Definition 3.*

Proof. Implied from the indistinguishability of XDHIES (Theorem 4).

Theorem 8. *(Unforgeability) The beacon notification protocol achieves unforgeability according to Definition 4, where H is modeled as a random oracle.*

Proof. Implied directly from the pseudorandomness of $\mathsf{Bcn}()$ and the randomness of H in the mapping table $\mathcal{M}$.

Theorem 9. *(Integrity) The beacon notification protocol achieves integrity according to Definition 5.*

Proof. Implied from the fact that the observer verifies with the cloud server that a received beacon's output is valid before using it.

Theorem 10. *(Correctness) The beacon notification protocol is correct according to Definition 6.*

Proof. We need to show that

$$(s, m) = \mathsf{Own}_{\mathsf{Svr}(\mathsf{Obs}(\mathsf{Bcn}_i(t,s),m)}(\mathsf{Obs}(\mathsf{Bcn}_i(t, s), m)).$$

Recall that $\mathsf{Bcn}_i(t, s) = pk$ where $pk = g^{r_i \cdot \mathsf{PRF}_{x_i}(t,s)}$. Thus

$$\mathsf{Obs}(\mathsf{Bcn}_i(t, s), m) = (H(pk), \overline{\mathcal{E}}_{pk}(m)),$$

and

$$\mathsf{Svr}(H(pk), \overline{\mathcal{E}}_{pk}(m)) = \mathcal{M}[H(pk)] = i.$$

Therefore, the i'th owner finds (t, s) from $\mathsf{Tbl}_i[H(pk)]$, computes $sk = r_i \cdot \mathsf{PRF}_{x_i}(t, s)$, uses it to decrypt $m = \overline{\mathcal{D}}_{sk}(\overline{\mathcal{E}}_{pk}(m))$, and output s, m.

9 Complexity of Time and Space

We next consider the complexity of each component in the beacon notification protocol:

Beacon: Beacon i computes the public key values, where a public key 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 [5]). 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.[1] 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 public key 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. The choice of a pseudo-random r_i is done via an extraction from a (possibly forward secure) pseudorandom generator (based on symmetric key operation) and a modular reduction in the field to get proper value from he drawn random long enough string.

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 ($g^{r \cdot \mathsf{PRF}_x(t,s)}$) which the observer gets from a near by beacon. Again, the fixed-base exponentiation can benefit significantly from [5], 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).

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

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 public key 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 public key 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 required size per beacon is 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. For efficiency and privacy the table values are not retained beyond the day.

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 public key values to the server. Using the above parameters, each entry of Tbl requires 128 bits, 32 bits and 2 bits for the public key value, time and signal, respectively. For 24 hours and public key value change every 20 minutes, the table is of size 72×162 bits $= 12$ KB. Computing Tbl entails $72 \times 3 = 216$ exponentiations for computing the required public key values. Each group exponentiation is with a fixed base and can therefore be computed using the efficient algorithm of [5]. 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 public key values to the cloud server in batches during the day).

To summarize, the beacon notification protocol is efficient in all the relevant parameters and can be easily integrated in real-world applications (in fact, parts of it have already been adopted to and implemented within a concrete setting).

10 Related Work

Beacons and BLE broadcasting in particular are widely useful in real-life proximity applications: contact tracing systems like Google-Apple Exposure Notification (GAEN) [3,8] has been such an example for a mobile-to-mobile signaling. We also note that there are works that study the beacons privacy such as [7].

However, we are not aware of any work with respect to the beacon notification problem as derived from the IoT setting and defined in this paper (essentially a beacon to owner real-time signaling via an observer), hence we consider below the closest protocols for different and weaker versions of this problem as is reflected by industrial deployments.

Specifically, in Samsung's SmartTag [13] and Tile [1] the beacon does not send a status so obviously no beacon's status security is provided. In addition, SmartTag and Tile do not provide end-to-end security with respect to the observer's message.

Apple's FindMy [12], on the other hand, provides end-to-end security with respect to the observer's message but

- without forward and backward security for all future/past messages,
- without beacon's status support,
- without efficient real-time communication,
- without observer's message integrity,
- and without formal cryptographic definitions, threat model, and security proofs.

It is worth noting that FindMy aims at neither an efficient real-time communication nor observer's message integrity, due to 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 [6,4] 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. In contrast, achieving anonymity in our server model relies on the server honestly erasing its data after using it in real time for association. Finally, we note that Apple has published neither a detailed description of the FindMy protocol nor cryptographic proofs of its security.

11 Conclusions

We presented the general problem for the IoT involving broadcasting beacons. In particular, we derived and presented novel cryptographic definitions for the beacon notification problem (including its security, privacy, and integrity requirements). To solve the problem we presented an extension for DHIES, which we called XDHIES and built upon it our beacon notification protocol. We then proved that our resulting protocol achieves all the formalized requirements.

Furthermore and looking forward, we believe that our new protocol, given its unique and comprehensive security features, its versatility, and its suitability for real-world IoT applications, may be considered for standardization for global use in the IoT area.

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. Apple, G. Privacy-preserving contact tracing. `https://covid19.apple.com/contacttracing`.
4. 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).
5. Brickell, E.F., G. D. M. K. W. D. Fast exponentiation with precomputation. In *Advances in Cryptology — EUROCRYPT' 92* (1993).
6. 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).
7. David, L., Hassidim, A., David, Y., and Yung, M. The battery insertion attack: Is periodic pseudo-randomization sufficient for beacon privacy? *Proceedings on Privacy Enhancing Technologies* (2025).
8. David, L., Hassidim, A., Matias, Y., and Yung, M. Scaling up gaen pseudorandom processes: Preparing for a more extensive pandemic. In *European Symposium on Research in Computer Security* (2022), Springer, pp. 237–255.
9. 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).
10. 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.

11. 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.
12. 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).
13. 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).