Ephemeral Identifiers:
Mitigating Tracking & Spoofing Threats to BLE Beacons

Avinatan Hassidim, Yossi Matias, Moti Yung, and Alon Ziv

Google Inc.

April 14, 2016

Abstract

Bluetooth Low Energy (BLE) beacons broadcast their presence in order to enable proximity-based applications by observer devices. This results in a privacy and security exposure: broadcast devices are typically susceptible to tracking and spoofing based on the IDs used by the beacons.

We introduce a scheme consisting of cloud-based Ephemeral Identifiers (EID) which allows only authorized parties to properly identify the beacons broadcast; it mitigates the basic tracking and security threats while keeping high utility. We outline a formal model of privacy which is obtained with our scheme, present its implementation within the Eddystone BLE format, and discuss possible extensions.

1 Introduction

Beacons based on the Bluetooth Smart (BLE) standard have recently been gaining acceptance as preferred enablers for proximity-sensitive experiences (see, e.g., [13]). Such beacons use BLE in a broadcast mode, repeatedly sending static "advertisement" packets, and thus requiring very little power (for protocol processing or for radio operation). This has the advantage that they can achieve long lifetimes while keeping to a small device size. Particularly interesting use-cases for such beacons include proximity beacons, used to enable location-based services, and object-tracking beacons, which enable tracking of portable physical objects.

The broadcast-only mode of BLE presents some security challenges. While the Bluetooth Specification [4] contains some security mechanisms, these are mostly aimed at protecting the data flow when paired devices are already connected [16]. When a device is operating in the broadcaster mode, the only security mechanism provided by the standard specification is use of random MAC addresses [4, vol. 3, § C.10.7.4]. To be viable, in many cases, broadcast-only applications will require additional application-level security for their data payloads. In addition, the standard random-address mechanism does not protect against replay attacks, a protection which is desirable for most security-sensitive applications.

We consider the following threats to be critical for users of beacons, in particular in applications which employ object tracking:

Unauthorized Tracking Beacon broadcast IDs are available to read by any nearby receiver. This enables many abuses, including after-the-fact unauthorized tracking.[1]

Forgery An adversary can forge the advertisements of any beacon broadcasting a consistent ID.

Showrooming Beacons are often used to identify microlocation. An adversary can use the beacon information of the owner to offer competing services to those given in that location.

In this paper we present a scheme consisting of ephemeral identifiers with cloud-based resolver service, we describe how such scheme can address the problem at scale, discuss theoretical foundations for our protocol, and analyze the design choices we have made from a pragmatic perspective.

[1] In after-the-fact tracking, an eavesdropper may use a widely– deployed mobile app to collect identifiers and their locations, and use this to later infer the beacon's historical location once learning the beacon's ID.

1.1 The Crux of Our Solution

Our solution relies on having a beacon advertize ephemeral identifiers (EIDs), rather than a static ID. The EIDs are generated using cryptographic means and can only be linked back to the beacon, or information associated with the beacon, by authorized users. Thus, the beacon's owner can identify the EID while for other devices it would look like a random ID. In addition, the owner of the beacon can register the beacon with a cloud based global *resolver*, which enables authorized non owners to identify the beacon (via the cloud). The protocol consists of the following components:

- Each beacon has a symmetric key by which it is identified. The key is known to the owner of the beacon.
- The identifying information in beacon advertisements consists of a deterministic pseudo-random function of the current time (with the correct precision) keyed with the beacon symmetric key. We call this value the "ephemeral ID". Note that this is essentially an encryption of time, with no additional salt.
- Whenever the owner of a beacon observes an advertisement, it can compute its own version of the ephemeral ID and compare it with the received ID. This way the owner of the beacon can identify the beacon, a process we call "local resolution of the ephemeral ID".
- The owner of the beacon can register the beacon with a trusted global *resolver* of ephemeral IDs. During the registration process, the owner sends the key shared with the beacon (or some agreed upon derivation of it) to the resolver.
- Any phone/device which observes an advertisement can transfer the observed data to the global resolver. The resolver then attempts to identify the beacon using all registered keys.[2] We call this process "cloud/global resolution of the ephemeral ID".
- The global resolver can notify the phone/device which observed the beacon and/or the owner with information derived from the identity of the beacon, in accordance with the corresponding sharing permission policy. Any such information is passed over secured channels (e.g. TLS).

A more detailed description and analysis is given below. This scheme serves as the basis for the Eddystone-EID specification and an implemented resolver.

1.2 Related Work

The Internet of Things, whose roots trace back to 1999 in the MIT Auto-ID Center [10], has from its beginning been concerned with portable objects. Initial projects focused on the use of Radio Frequency ID (RFID) technology, and these led to research on the security and privacy aspects of tag tracking. In fact, despite the challenges of power consumption, standardization, processing capabilities etc., perhaps the biggest challenges for adoption of the Internet-of-Things are in the security and privacy area [6].

There have been numerous investigations for privacy and security of RFID, some of which were RFID mechanism and device specific (like shielding the token, etc.), and some suggested a use of a pseudonym, which is the first natural idea in hiding actual identity and may be effective for exact identity association but does not solve the tracking problem.

Other solutions suggested replacing the advertised value during every interaction between an RFID reader and a tag. [17] originally suggested cutting off the identifying bits, and only passing the generic identity part as a direction towards anonymous identity. In [12] the authors suggested – in the context of book identification in libraries – assigning a temporary ID in each interaction of the library with a book that is loaned, this way preventing linking based on identity tracking in the library, while assigning a permanent ID back when a book is returned. [8] finally suggested having a set of IDs assigned temporarily (random ID) and updating the ID cryptographically between interactions with the readers, and a method to synchronize successful readings among the set of readers. Obviously the set of few readers can be trusted much more than any reader on the cloud, and synchronizing readings is possible at that scale (while being prohibitive in a cloud scale). Note that this random identification has motivated various

[2] Note that the resolver does not iterate over all possible keys (which is a prohibitively large space), but only over legitimate keys that were registered.

works that use cryptographic tools and encryptions for performing the task: [9] suggested using the system hosting the tag to perform the encryption with a public-key operation. Golle et al. [7] defined universal re-encryption, which allows a reader to re-encrypt the identity of a tag without knowing either the plaintext or the public key it uses. However, this scheme doubles the size of the ciphertext. Ateniese, Camenisch, and de Medeiros describe a similar solution based on elliptic curve cryptography, which also allows verifying message integrity [2].

A different type of proposal (in which the tag needs to compute) was presented by Weis et al. in [19], which suggested a MAC based random computation on a random challenge (and a shared key). Similarly, in [11] Molnar, Soppera, and Wagner suggested a rotation of pseudonym based on a PRF computed over a random nonce or a counter, where an interaction with the reader can assure the computation is done with the actual key.

Finally, it is widely accepted in the industry that beacons pose a new security threat [18]. Kontakt.IO propose to shuffle some of the fields in the advertisment (called "Major" and "Minor") in order to prevent forgery, but they do not prevent tracking [3]. Estimote has a solution which also rotates the UUID, but relies on secure access to the estimote cloud for beacon identification, even for the owner of the beacon [1]. The technical details of these methods were not fully disclosed.

1.3 Organization

The rest of the paper is organized as follows. In Section 2 we describe the theoretical model and results. In Section 3 we present the technical requirements, and the protocol. In Section 4 we discuss extensions, and known weaknesses. In Section 5 we present the reasons for the design choices that we made. Finally, Section 6 presents our conclusions.

2 Model and Definitions: Ideal Model of Identity Advertisement and Security

We model the device as having access to a Pseudorandom Function (PRF) as a basic primitive, representing the value of the cryptographically generated MAC address and encryption of a counter.

A pseudorandom function family $F_k()$ returns values $F_k(x)$ for a given parameter x. The function is keyed with a key k in $\{0,1\}^n$ (where n is the security parameter), so there are exponentially many functions in the family.

$F_k()$ is called pseudorandom (PRF) if the following experiments hold:

- The challenger either chooses a random function F from the family of functions, or a truly random function (imagine: a huge table of random values) R.
- The adversary is allowed to ask a polynomial number of queries x and gets the answer which is the function applied to x.
- The adversary guesses a bit $b = 0$ for random and $b = 1$ for pseudorandom.
- We say that the function family is indeed pseudorandom if the advantage of the adversary is negligible (the guessing probability is better than $\frac{1}{2}$ by an amount at most a super inverse polynomially small in the function size parameter).

How do we define privacy for a collection of devices, each holding an instance from a pseudorandom family? We assume an ordered collection of pseudorandom functions (PRFs) are given as a blackbox access to the adversary: $F^1_{k_1}, F^2_{k_2}, \ldots F^m_{k_m}$ where each $k_i \in \{0,1\}^n$ is the key of the i'th PRF. The adversary is given the outputs of this ordered set for a while (modeling tracing of the devices and their outputs), it then loses tracing and the set is permuted within a distribution of permutation, at that point the adversary needs to correlate the permutation with the original order: this is a linkability task, perhaps the weakest privacy requirement, where the devices are blended in the crowd. The adversary should not learn more about the permutation than what his a priori knowledge about the permutation distribution is (this is a type of "semantic security" for device permutation space).

What we will show formally is that linkability of the type above in fact leads to a distinguisher over one of the PRFs, contradicting the cryptographic assumption on which the scheme is built upon.

Note that by considering any distribution on permutation, we embed in the distribution the adversary's partial knowledge of it (e.g., if some devices do not permute, or otherwise remain known/ tracked, the distribution encompasses this).

Next we show how indistinguishability implies that the attacks above (technically presented) are blocked.

We say that a *distinguisher* $\mathcal{D}$ is an algorithm that takes as input a distribution over permutations $\mathcal{P}$ of m elements, and a set of PRFs $F^1_{k_1}, F^2_{k_2}, \ldots F^m_{k_m}$ where each $k_i \in \{0,1\}^n$ is the key of the i'th PRF, and takes part in the following protocol, where t is some polynomial of n and m (in fact, m itself in practice is polynomial in n):

1. $\mathcal{D}$ chooses some values $x_1 \ldots x_t$.
2. For each $1 \leq s \leq t$, and for each $1 \leq i \leq m$ the distinguisher $\mathcal{D}$ gets to see $F^i_{k_i}(x_s)$.
3. $\mathcal{D}$ chooses some value x which was not chosen so far.
4. A random permutation π is sampled according to the distribution $\mathcal{P}$.
5. $\mathcal{D}$ is permitted to see $F^{\pi(i)_{k_{\pi(i)}}}(x)$ for every i.
6. $\mathcal{D}$ needs to identify π.

Clearly, $\mathcal{D}$ can just choose the most probable permutation in $\mathcal{P}$, which will give a success probability equal to $\frac{1}{\delta}$, where δ is the min entropy of $\mathcal{P}$.

We shall use $\mathcal{D}(\mathcal{P}, F^1, F^2, \ldots F^m)$ to denote the successful run of the distinguisher $\mathcal{D}$.

Theorem 1. *If there exists a distribution over permutations $\mathcal{P}$ and a distinguisher $\mathcal{D}$ such that the success probability of $\mathcal{D}$ is at least $\frac{1}{\delta} + \epsilon$, where δ is the min entropy of $\mathcal{P}$ and ϵ is inversely polynomial in n, m, then $\mathcal{D}$ can be used to distinguish a PRF from a truly random function by looking at a polynomial number of inputs.*

Note that our definition is cast in the language of semantic security, where all the knowledge Eve has (e.g. controlling some of the beacons) is modeled with $\mathcal{P}$, and Eve can not extract any more information than what she already has.

Proof. Suppose for contradiction that $\mathcal{D}$ and $\mathcal{P}$ exist, such that for every m pseudo random functions, $\mathcal{D}$ succeeds in identifying the permutation with probability at least $\frac{1}{\delta} + \epsilon$.

The proof is done using a hybrid argument. Let $G_1, \ldots G_m$ be truly random functions. Then clearly $\Pr(\mathcal{D}(\mathcal{P}, G_1, \ldots G_m)) \leq \frac{1}{\delta}$, since when the functions are random $\mathcal{D}$ can just output a permutation from $\mathcal{P}$ (intuitively speaking, a truly random value cannot really be attributed to one random function and not another).

However, let us look at hybrid vectors of prefix of the vector of functions being pseudorandom and the suffix of truly random ones (for each prefix length); the above means that there is i such that

$$\Pr(\mathcal{D}(\mathcal{P}, F^1 \ldots F^i, G_{i+1}, \ldots G_m)) - \Pr(\mathcal{D}(\mathcal{P}, F^1 \ldots F^{i-1}, G_i, \ldots G_m)) > \frac{\epsilon}{m}$$

Now, given a function F and the task to establish if it is pseudorandom or not, we can just use $\mathcal{D}$ to find out, by choosing $i-1$ PRFs $H_1 \ldots H_{i-1}$ for which we know the key, and $n-i$ random functions (in fact we do not need the entire table of the random function but just its values on $x_1, \ldots x_t$). Running $\mathcal{D}$ with $\mathcal{P}$ and the functions: $H_1 \ldots H_{i-1}, F, G_i, \ldots G_m$, we can measure its success probability, up to accuracy $\frac{\epsilon}{2n}$. If it's closer to the success probability when there are i PRFs, then F is a PRF, while if it's closer to the success probability where there are $i-1$ PRFs then F is truly random. We have just used $\mathcal{D}$ to distinguish between a PRF and a random function, which contradicts the assumption.

□

Note that our impossibility result admits the following corollaries:

- Given a promise that either π_1 or π_2 are applied, each with probability $\frac{1}{2}$, no algorithm can identify which permutation was used with polynomially small advantage.

- No algorithm can distinguish between the identity permutation and any other permutation with probability better than $\frac{1}{2}$ plus any inverse polynomially small factor (where the identity occurs with probability $\frac{1}{2}$).
- If Eve is trying to follow Alice, and Eve controls s out of m other beacons in the vicinity, Eve's chances of guessing who is Alice are only negligibly better than $\frac{1}{(m-s)}$.

3 Solution Design Realization: Implementing the Ideal Model

3.1 Realization: the Cryptographic Primitives

The following sections all use a simple cryptographic scheme, built mostly around use of AES-128 in various modes and mode combinations. AES is chosen here due to its high ubiquity in Bluetooth devices (since AES is already used in parts of the Bluetooth specification), as well as the high level of confidence in its security. For extra efficiency, AES is used almost solely in one direction (encryption), either as a plain pseudorandom function (PRF) or in an authenticated-encryption mode (EAX).

Each data field is protected using a different scheme and a different key; however, a simple implementation can use the Bluetooth IRK (identity resolution key) for almost all schemes. This key already exists in the Bluetooth specification, and is reused here (in safely-different modes) to ensure even the simplest Bluetooth devices are able to implement our protocol.

When the use case requires many devices to recognize the beacon (e.g. object-tracking tags, proximity beacons), the relevant key (or keys) will be shared between the beacon and the resolver. If the beacon is to be used by just a single mobile device (e.g. device-unlock tokens), the key may be shared only between this device and the token (eliminating the functionality of the resolver and making this essentially a pseudonym).

3.2 Different forms of identity

In order to present a solution to the security and privacy threats on BLE beacons, we should protect three types of data:

1. Bluetooth MAC addresses
2. Application-level device identifiers
3. Device-specific broadcast mutable state

We will discuss each of these separately.

3.2.1 Hiding Bluetooth MAC Addresses

The Bluetooth specification provides a mechanism for devices to randomize their MAC addresses, described in volume 3, part C, section 10.8.2 of the Bluetooth 4.2 specification. We choose to use resolvable private addresses (RPA), where the broadcaster periodically chooses a new 22-bit random nonce (termed prand) and expands it to a 48-bit address using encryption with the IRK. The resulting addresses look fully random, but are easily resolved by an observer with access to the IRK.

Depending on the use-case, the observer device may have direct access to the IRK; in such cases, the observer just needs the MAC address in order to identify the broadcaster. Other use-cases (such as proximity beacons) will not distribute the IRK to observer devices, and will not be able to directly use the randomized MAC address for beacon identification.

Due to the specific scheme used by the Bluetooth specification, RPA values are susceptible to replay attacks. It is therefore not recommended for any application to use RPA resolution as its sole means of identifying a broadcaster.

3.2.2 Hiding Application-Level Identifiers

Our proposal does away completely with existing identifiers and replaces them by a different (and non-traceable) ephemeral identifier (EID). This EID can be resolved by anyone with access to the ephemeral ID key (EIK); unlike the RPA, the EID values are fully predictable, so a backend that needs to resolve millions of identifiers per second can pre-generate a lookup table from EID to true ID. This same predictability also makes EID values safe against replay attacks.

In this proposal, the EID is generated by encrypting a counter of seconds since a predefined "epoch".[3] In order to conserve battery power, the EID is computed only once per "tick", which is 512 seconds by default. This does permit some tracking for short time frames (one tick). Note that, in order to save space in broadcasts, the generated identifier is truncated to 64 bits.

3.2.3 Hiding Extra Properties

Extra properties are used by some beacons to indicate data about the beacon hardware itself (telemetry), or data collected from the outside world by on-device sensors (mutable state). In the case of telemetry, the device can choose the collection period by itself; however, other sensors may impose their own timings. An example sensor case is a bracelet that serves as an unlock token: the extra property in this case is a counter of bracelet open/close events, which needs to remain equal to the last observed value in order to perform the unlocking.

Since extra properties fields can change on their own schedule, the encryption scheme needs to introduce additional variability when encrypting them. These fields are therefore encrypted as follows:

- Generate a 48-bit nonce, consisting of
  - The common time base.
  - 16 random bits.
- Encrypt the state signal using AES-EAX, using the nonce computed in the previous step and the EIK, and producing a 16-bit authentication tag.
- Transmit the 16 random bits, the encrypted state, and the 16-bit authentication tag.

Decryption of this information will happen only after the EID is resolved, and hence the key is known.

3.2.4 Synchronization

It is important that the EID and extra properties fields change at the same time as the MAC address, as if the MAC address changes and another field (e.g., the EID) stays the same, a persistent attacker can correlate the new MAC address with the old one.

Likewise, whenever the EID changes, the MAC address and extra properties must be recomputed.

Since beacons' clocks are never synchronized after the initial provisioning, they are likely to accumulate some "drift" during their operational lifetime. Observers (and backends) must be able to accommodate up to 500 ppm of such drift.

Additionally, in order to prevent attackers from recognizing a beacon using the exact time it changes identities (using its drift as a "signature"), beacons can choose a random point within each tick at which to change their identifiers. This randomization has to be independent for each tick—i.e., the ticks must start at a correct time relative to the epoch, regardless of when previous identifiers were exposed.

3.3 Global Resolution of Ephemeral IDs

When an observer received an unknown ephemeral ID, it queries the global resolver. This is the case, for example, when collecting crowdsourced data for object tracking (as the observers have no relation to the tracked objects). It may also be the case for proximity beacons, which do occasionally move.

In this mode of resolution, since the variable component (time) is also known by the backend, it is able to predict the EID values for all broadcasters at all times. It is thus able to pre-generate a mapping from all expected EID values (in the coming time period) to stable ID. When the backend receives a

[3] The epoch time has been arbitrarily chosen as 2015-01-01 midnight UTC.

report using an ephemeral ID, it will use it to locate the corresponding stable ID and continue processing with this identity. Note that since no nonce is used in generating the EID, a single mapping can be used for all possible observations for the period of a single "tick".

The design principle here is that the global resolver does not need to perform any cryptographic operations at query time—all cryptographic operations (computing expected EID values) can be performed offline, and the amount of computation can be traded against storage size. A resolver supporting 30 million beacons will need a database of 480 megabytes per "tick" it can decode; if we assume 512-second "ticks", this translates to 6.75 gigabytes for four hours' worth of EID values. Another 2.4 gigabytes would account for 60 days' worth of accumulated drift, so this case can be easily supported purely out of RAM on a modern server.

To accommodate larger numbers of beacons (e.g. 10 billion, to be on the same scale as Gartner's estimates for 2020 growth [15]), the data set can be sharded (distributed) across multiple servers, with each server handing some prefix of the EID bits.

4 Known Limitations and Possible Extensions

The proposed protocol has some limitations that we need to be aware of, and it does not mean to bulletproof any possible issue which may arise in the field or in the implementation stage.

First and foremost, if it is not widely adopted, or a single beacon is using these ephemeral ids in an area where no other beacons are using them, then tracking would be possible. This is inherent limitation of any solution of the type of "hiding in the crowd."

In addition, we chose not to rotate the broadcast message every transmission, and left the exact interval as a parameter (we expect that it will be used in the order of 512 seconds, or around ten minutes). This allows two attacks:

- A beacon can be followed around for a short period of time.
- If beacons change their broadcast message sequentially, it is still possible to follow them, as long as just one beacon is changing the message every time, and Eve has access to all the broadcasts sent by each beacon. If Eve only gets a snapshot of the situation once in a few minutes, and several beacons changed their broadcast at that time, we are safe again.

Finally, erroneous implementations of the Eddystone specification can leak information, for example, if a beacon not rotate the MAC simultaneously with the broadcast then the PRF notion is not really realized and this allows tracking.

We present some extensions to the protocol. Some of the extensions we present were already implemented in the Eddystone BLE format from Google.

4.1 Extension: Forward Secure Ephemeral IDs

In the current scheme, if Eve learns Alice's key, and Eve recorded past BLE advertisements, Eve can learn information about Alice's past whereabouts. One way to eliminate this threat is to apply a one way function on the key Alice uses whenever the timestamp field changes. For example, one can simply use $\mathrm{EIK}_{t+1} = \mathrm{SHA256}(\mathrm{EIK}_t)$ (under the assumption that the function is emulating a random oracle, the new key is random, and the old key is protected against an attacker which gets hold of the current key). If SHA256 is not supported, one can even use $\mathrm{EIK}_{t+1} = \mathrm{AES}_{\mathrm{EIK}_t}(\mathrm{EIK}_t) \oplus \mathrm{EIK}_t$. Note that we do not require a cryptographic hash function here, any OWF is enough.

We did not implement this in the Eddystone BLE format, as it complicates the implementation for equipment manufacturers.

4.2 Extension: Key Revocation

It is possible to use several resolvers to identify a given beacon (e.g., a local resolver managed by the owner and a global one). A naive way to do this is to share the EIK between all the resolvers. However, the owner may want to give a revocable key to the global resolver. It is theoretically possible for the

owner to compute every day a list of EIDs for that day, and send the list to the global resolver, thus bypassing the need to send the key to the global resolver. However, this does not scale well.

An alternative solution which appears in the Eddystone BLE format is to use two levels of EIK. Each day, we generate a daily EIK by computing a one-way hash of the number of days since the epoch and the EIK.[4] We then compute all the EIDs using the daily EIK, instead of using the "real" EIK. This is far more scalable, as the owner of the beacon can compute several daily EIKs every few days and send them to the global resolver.

4.3 Extension: Negotiating EIK

In this paper we didn't consider the question of how to negotiate the EIK, treating it as a given. Two simple ways one can consider are:

1. Use ECDH between the token and the phone or ECDH between the beacon and the cloud. One disadvantage of this system is that the beacon needs to support ECDH, which may not always be the case.

2. Hardcode the EIK during production, and send it also to the cloud. This is problematic, since the factory may keep logs of the keys, it is possible that Eve will get the keys en route to the cloud, etc.

We prefer the first way, and point to the specifications [5] for more explanations, as this is not the focus of this work.

4.4 Extension: Offline Resolution by Non-Owners

Occasionally, mobile phones/ devices need to efficiently detect when they are near an "interesting" beacon.[5] Ideally, such detection needs to happen even when the mobile device is offline.

In order to enable this, the mobile device can ask the global resolver to produce offline resolution data about the device's area (or areas it is about to visit, using some heuristics).

The offline resolution data is built as

1. Choose a random key I.

2. Build a list of "interesting" beacons in the area the device specified.

3. For each of these beacons, compute its expected ephemeral IDs for the near future (e.g. the next hour).

4. Use a KDF to expand each ephemeral ID to a 128-bit key, then use this key to encrypt the beacons's static metadata using a nonce-based AEAD scheme.

5. Use HMAC_I to convert the ephemeral ID to an identifier for its associated encrypted block.

When the mobile device now observes an ephemeral ID e, it will compute $\text{HMAC}_I(e)$ and use it to locate the associated encrypted data block. If such a block is found, and decrypts successfully, the device now has the data it required.

5 Design decisions

In this section we describe the reasons behind the design decisions made in the system, and further articulate the reasons for our new design given the above related work in the RFID domain.

[4] This is similar to a masterkey system cryptography, which derives the daily key from the day and the master key (the EIK).

[5] A beacon is "interesting" if any app on the device has behaviours triggered by this beacon.

5.1 Advertisement is Determined by Key and Time

A natural scheme is for the beacon to choose a nonce, encrypt it, and advertise the nonce as well as the encrypted nonce for identification. Such a scheme would not enable global resolution: assuming the resolver has a million registered beacons, and it is asked to resolve an advertisement for each every 10 minutes, it is not feasible to go over all registered keys every time the global resolver is queried. On the other hand, as we explained above, going over all registered keys once every 512 seconds and performing a lookup on every call to the resolver is feasible even if there are billions of registered keys.

5.2 Our Solution vs. the Available RFID Protocols

We adopt the concept of cryptographic randomization (throttling) of the identity as a natural idea, but there are several important differences between the RFID and the IoT settings:

- The IoT beacons are pure broadcast (no interaction with a reader), and no synchronization among observers while the set of observers is global.
- The IoT beacon has more computation power than an RFID, and can maintain a reasonably-accurate clock.
- BLE range is larger, and any cell phone can act as a BLE sniffer, so it's easier for the attacker.

5.3 Utility of a Global Resolver

One can consider a scheme in which each beacon has a key, known only to its owner. The beacon will then advertise an encryption of its identity and some random nonce, and the nonce itself. While such a system can prevent tracking, its utility is limited. In particular, it is inferior to a system with a global resolver in the following settings:

1. **Crowdsourcing without tracking:** Suppose we want Alice to know where is her fitness device, although it was observed by Bob's phone and not her own. If Bob passes all the advertises he receives to the global resolver, the resolver can identify the beacons Bob see, and notify their owner.
2. **Beacon deployment without showrooming:** Consider again the case where Alice writes app which requires microlocation, and deploys beacons to support the app. The global resolver would help Alice's users know where they observe one of her beacons, but will only transfer this information to Alice's app (e.g. via the Google Cloud Platform Proximity Beacon API [14]). Eve can write a competing app, but the resolver will not pass the beacon information to Eve's app.

5.4 Public Key Encryption of the Identity

Another possible scheme was for each beacon to hold the public key of the global resolver, and then each beacon would broadcast the encryption of its identity and some random nonce. There are three main reasons for not choosing a public key based solution

- Public key encryption is expensive in terms of battery, and may not even be supported on some beacons. We can not rely on frequent encryption.
- Public key encryption requires long messages, and a BLE broadcast is short (31 bytes including some headers)
- In the case where Alice is carrying a fitness item, she needs to be able to identify her item without getting to the cloud (and clearly Alice should not know the private key of the cloud). An ideal solution can be adaptively used for local identification as well as cloud-mediated identification, and key sharing with owners solve this issue.

6 Conclusion

The basic standard and common practice of BLE beacon systems lacks adequate layers of security for all applications. Users may be walking around with BLE devices, which broadcast their identity with a range of up to 100 meters. Further, harvesting these identifiers is very easy, via a phone, or simple BLE sniffers. Ephemeral IDs address this privacy challenge, while also providing an additional layer of security – a user can see a beacon for the first time and know what this beacon represents, but at the same time cannot forge this beacon to another user.

To enable these applications we pass the entire smartness to the cloud, using a trusted third party as a global resolver. Doing this requires the resolver to be able to serve billions of encrypted requests from unidentified sources. Since the beacons can not afford to encrypt using the resolver's public key, the resolver needs to try all the keys it has, and we have shown how this can be done in an efficient manner by applying deterministic encryption of time. Moreover, we have shown that the beacons can afford the computation required by the protocol.

We augment our system analysis with an abstraction of the problem, and prove that the protocol adheres to a definition of anonymity which is similar in spirit to semantic security.

Finally, the system we presented is the basis for an open specification [5] published in GitHub within the Eddystone open beacon framework from Google with corresponding resolver implementation from Google Cloud Platform's Proximity Beacon API, demonstrating the feasibility of the design choices we have presented.

Acknowledgements

We would like to thank Arnar Birgisson, Thai Duong, Bryan Horling, Nirdhar Khazanie, Matthew Kulick, Peter Lewis, Stefano Maggiolo, Jonathan Morace, Peet Sasaki, Reuven Sayag, Marius Schilder, Jessica Staddon and Chandu Thota for their helpful contributions.

References

[1] *Are Estimote Beacons secure? How does Secure UUID work?* URL: `https://community.estimote.com/hc/en-us/articles/201371053-Are-Estimote-Beacons-secure-How-does-Secure-UUID-work-` (visited on 04/10/2016).

[2] Giuseppe Ateniese, Jan Camenisch, and Breno de Medeiros. "Untraceable RFID tags via insubvertible encryption". In: *Proceedings of the 12th ACM conference on Computer and communications security*. ACM. 2005, pp. 92–101.

[3] *Beacons Have Been Vulnerable For Too Long. It's Time We Fixed It.* URL: `http://kontakt.io/blog/beacon-security/` (visited on 04/10/2016).

[4] Bluetooth SIG. *Specification of the Bluetooth® System*. Version 4.2. 2014.

[5] *Eddystone-EID*. URL: `https://github.com/google/eddystone/eddystone-eid` (visited on 04/11/2016).

[6] Federal Trade Commission staff. *Internet of Things: Privacy & security in a connected world*. 2015. URL: `https://www.ftc.gov/system/files/documents/reports/federal-trade-commission-staff-report-november-2013-workshop-entitled-internet-things-privacy/150127iotrpt.pdf` (visited on 04/11/2016).

[7] Philippe Golle et al. "Universal re-encryption for mixnets". In: *Topics in Cryptology–CT-RSA 2004*. Springer, 2004, pp. 163–178.

[8] Ari Juels. "Minimalist cryptography for low-cost RFID tags". In: *Security in Communication Networks*. Springer, 2004, pp. 149–164.

[9] Ari Juels and Ravikanth Pappu. "Squealing Euros: Privacy Protection in RFID-Enabled Banknotes". In: *Financial Cryptography: 7th International Conference, Revised Papers*. Springer Berlin Heidelberg, 2003, pp. 103–121.

[10] *MIT Auto-ID Center*. URL: `http://autoidlabs.org/wordpress_website/` (visited on 04/06/2016).

[11] David Molnar, Andrea Soppera, and David Wagner. "A scalable, delegatable pseudonym protocol enabling ownership transfer of RFID tags". In: *Selected Areas in Cryptography*. Springer. 2005, pp. 276–290.

[12] David Molnar and David Wagner. "Privacy and security in library RFID: Issues, practices, and architectures". In: *Proceedings of the 11th ACM conference on Computer and communications security*. ACM. 2004, pp. 210–219.

[13] *Nearby Messages API for Android: Get Beacon Messages*. URL: `https://developers.google.com/nearby/messages/android/get-beacon-messages` (visited on 04/10/2016).

[14] *Proximity Beacon API Overview*. URL: `https://developers.google.com/beacons/proximity/guides` (visited on 04/11/2016).

[15] Janessa Rivera and Rob van der Meulen. *Gartner Says the Internet of Things Installed Base Will Grow to 26 Billion Units By 2020*. 2013. URL: `http://www.gartner.com/newsroom/id/2636073` (visited on 04/05/2016).

[16] Mike Ryan. "Bluetooth: With Low Energy Comes Low Security." In: *WOOT*. 2013.

[17] Sanjay E. Sarma, Stephen A. Weis, and Daniel W. Engels. "RFID systems and security and privacy implications". In: *Cryptographic Hardware and Embedded Systems-CHES 2002*. Springer, 2002, pp. 454–469.

[18] *The Hierarchy of IoT "Thing" Needs*. URL: `http://techcrunch.com/2015/09/05/the-hierarchy-of-iot-thing-needs` (visited on 04/10/2016).

[19] Stephen A. Weis et al. "Security and privacy aspects of low-cost radio frequency identification systems". In: *Security in pervasive computing*. Springer, 2004, pp. 201–212.