Acta Cybernetica **25** (2021) 319–349.

Lamred: Location-Aware and Privacy Preserving Multi-Layer Resource Discovery for IoT*

Mohammed B. M. Kamel[abcd], Peter Ligeti[ae], and Christoph Reich[bf]

Abstract

The resources in the Internet of Things (IoT) network are geographically distributed among different parts of the network. Considering huge number of IoT resources, the task of discovering them is challenging. While registering them in a centralized server such as a cloud data center is one possible solution, but due to billions of IoT resources and their limited computation power, the centralized approach leads to some efficiency and security issues. In this paper we proposed a location-aware and privacy preserving multi-layer model of resource discovery (Lamred) in IoT. It allows a resource to be registered publicly or privately, and to be discovered with different locality levels in a decentralized scheme in the IoT network. Lamred is based on structured peer-to-peer (P2P) scheme and follows the general system trend of fog/edge computing. Our model proposes Region-based Distributed Hash Table (RDHT) to create a P2P scheme of communication among fog nodes. The resources are registered in Lamred based on their locations which results in a low added overhead in the registration and discovery processes. Lamred generates a single overlay and it can be generated without specific organizing entity or location based devices. Lamred guarantees some important security properties and it showed a lower latency comparing to the centralized and decentralized resource discovery models.

Keywords: resource discovery, DHT, IoT

*This research has been partially supported by Application Domain Specific Highly Reliable IT Solutions project which has been implemented with the support provided from the National Research, Development and Innovation Fund of Hungary, financed under the Thematic Excellence Programme TKP2020-NKA-06 (National Challenges Subprogramme) funding scheme, by ÚNKP-20-3 New National Excellence Program of the Ministry for Innovation and Technology from the source of National Research, Development and Innovation Fund, by SH program and by the Ministry of Science, Research and the Arts Baden-Württemberg Germany

[a]Eotvos Lorand University, Budapest, Hungary
[b]Hochschule Furtwangen University, Furtwangen, Germany
[c]University of Kufa, Najaf, Iraq
[d]E-mail: mkamel@inf.elte.hu, mkamel@hs-furtwangen.de, ORCID: 0000-0003-1619-2927
[e]E-mail: turul@cs.elte.hu, ORCID: 0000-0002-3998-0515
[f]E-mail: christoph.reich@hs-furtwangen.de, ORCID: 0000-0001-9831-2181

DOI: 10.14232/actacyb.289938

1 Introduction

The Internet of Things (IoT) network consists of billions of resources distributed in different parts of the network. The huge number of resources and their different levels of accessibility (e.g. private resources, local resources and public resources) make the task of registering and discovering them a challenging task. Adopting a centralized scheme such as relying on a cloud service helps organizing the resources in an entity that has a high computation capability and can be used to discover those registered resources. But, in systems that rely only on a centralized entity a significant amount of traffic has to be used for the registration and discovery processes which might affect the overall efficiency of the system. Comparing to cloud computing infrastructure that send the traffic to a centralized cloud data center, the fog/edge nodes in the fog and edge computing infrastructures try to distribute the data among nodes and keep it as close as possible to the origin source of data. Hence, fog computing extends the cloud computing to the edge of the network, close to the point of origin of the data [3]. Processing the data locally during the registration and discovery of resources helps to achieve scalability, at the same time mitigates the potential privacy and security risks against single point of attack and failure. However, there should be a unique decentralized scheme that defines and arranges the relationship between the fog/edge nodes and their responsibilities.

Distributed Hash Table (DHT) creates an overlay by assigning a seemingly unique identifiers to the participating nodes. The generated overlay can be used to organize the distributed nodes in the decentralized resource registration and discovery processes [15]. The identifiers in DHT are generated by feeding some of the parameters of the peer nodes (e.g. IP addresses) to a hash function, and the output is used as the identifiers of the nodes. Depending on the identifier, each node is resided in a specific location in the overlay with a predefined responsibilities. Due to the random-looking behaviour of the hash functions, the output of the relatively close parameters in the input range might not be close in the hash space. While this property is required to ensure the random and uniform distribution of nodes and the stored data in the overlay, but adopting the original DHT technique in fog and edge computing infrastructures might results that two adjacent nodes reside in two far locations in the overlay. As a result, while adopting DHT in resource discovery [15] removes the centralized entity, but might map the geographically close nodes to distant nodes in the resulted space. If the nodes in the resource discovery models are distributed without considering their physical locations, an efficiency issue might be raised. This is due to the reason that the logical path of nodes on the underlying network could vary from the logical based path in the overlay network that is organizing the distributed nodes. Thus the lookup latency can be high, which in this case leads to operational inefficiency in applications running over it [18]. During organizing nodes in the resource discovery model, the locations of nodes have to be taken into consideration. Afterward, a resource is registered based on its location in a close node in the distributed system which reduces the required time to register and reach that specific node.

Therefore, while adopting DHT as a structured Peer-to-Peer (P2P) scheme to organizing fog and edge nodes in IoT has some advantages such as scalability and functionality without involving any centralized entity, but DHT might cause the data to be stored in a far node. In this paper we proposed a location-aware and privacy preserving multi-layer model of resource discovery (Lamred) in IoT. Lamred aims to keep the data as close as possible to the origin of the data by taking into consideration the locations of both resources and IoT gateways and utilizing a single DHT overlay. It can be implemented without specific location based devices, and add no extra local overhead comparing to traditional DHT overlays. Here are the main contributions of this paper:

- Propose Lamred, a new DHT based model as a P2P overlay for resource discovery in the IoT Network.
- Propose a Region based Distributed Hash Table (RDHT) for location aware resource registration and discovery. Lamred keeps the resources as close as possible to the clients, hence reducing the required time during the registration and discovery processes.
- Propose a private tag generation method in Lamred for private resource registration and discovery.
- Use cryptographic primitives to protect the private resources in the system and ensure the required anonymity and privacy in Lamred.

The rest of this paper is organized as follows. The next section defines some of the preliminaries. Section 3 summarizes the efforts in current research field of resource discovery. Section 4 describes Lamred, the proposed model of resource discovery, and introduces its different components. In Section 5 we evaluate the model, proof the required security properties and discuss the performance of Lamred. Finally, we conclude our work in Section 6.

2 Preliminaries

2.1 Cryptographic Primitives

Definition 1 (collision-resistant one-way hash function)**.** *A function H(.) that maps an arbitrary length input M into a fixed-length digest d is called collision-resistant one-way hash function it satisfies the following properties:*

- *Given M, it is easy to compute $H(M)$.*
- *Given d, it is hard to find any M s.t. $d = H(M)$.*
- *Given $d = H(M)$ and M, it is hard to find M' s.t. $M' \neq M$ and $H(M) = H(M')$.*

- *It is hard to find two distinct messages M' and M'' s.t. $M' \neq M''$ and $H(M') = H(M'')$.*

If the solution can be computed in the polynomial time, therefore it is considered *easy* to compute. On the other hand, if there is no solution known to solve the problem in polynomial time, it is considered a *hard* problem [7].

Definition 2 (Probabilistic Polynomial Time)**.** *An algorithm $\mathcal{A}$ is a PPT (Probabilistic Polynomial Time) if its probabilistic and $\exists c \in \mathbb{N}$ such that $\forall x, \mathcal{A}(x)$ halts in $|x|^c$ steps.*

2.2 Distributed Hash Table

DHT is a distributed system that creates a structured P2P overlay in a network. The participating nodes in the network can join and leave the DHT at any specific time. Upon joining a new node, a new identifier is assigned to it and depending on the assigned identifier, it will be responsible of storing a set of data in the network. Using multiple replicas helps the DHT to be fault tolerant and improves the availability of data in the network. Using identifiers instead of other types of addressing (e.g. IPs) helps to balance and manage the data storage among participating nodes without any centralized entity. In addition to load balancing, it solves the scalability by providing the service of generating the identifiers by the participating nodes themselves. There are several protocols to implement DHT such as Chord [29], Kademila [20], Pastry [28], and Tapestry [33].

DHT uses a large address space of integer numbers. The size of the address space depends on the fixed output size of the function that is used to generate the identifiers. The size of the key space is he same as the address space, i.e. the same function is used to generate identifiers for nodes and keys for the stored data. To achieve the random function of identifier generation and uniform distribution of data among all participating nodes a collision-resistant one-way hash function (definition 1) is used in DHT.

Similar to hash tables [19], the data in DHT is stored in key/value pairs. The value parameter includes any stored information about the data (e.g. the address of the data) and can be retrieved from the DHT based on its associated key. The key parameter of the key/value pair is generated by feeding specific information (e.g. the attribute of the stored data such as its name, its type, etc.) to the collision-resistant one-way hash function which produces a uniformly distributed randomized hash value. The generated hash value which represents the key parameter in the key/value pair is used to determine the responsible node in the network of storing this specific pair. To achieve the distributed indexing, DHT defines a specific portion in the key space that each particular node is responsible for. DHT has two implementation interfaces for storing and lookup: *Put* and *Get*. The *Put* interface takes the key/value pair and stores this pair in the DHT. The *Get* interface takes a single parameter key and lookup in the DHT to retrieve the identifier of a node that is responsible to store the corresponding value to the given key. In the DHT the

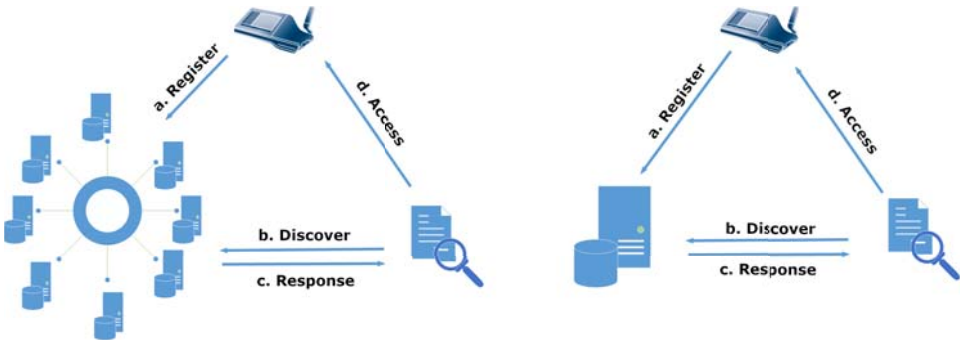
store (i.e. put interface) and lookup (i.e. get interface) operations are guaranteed to be done with an upper bound of $O(log(n))$, in which n is the number of nodes in the DHT. This feature guarantees that any participating node in DHT can store a pair of key/value or lookup based on a given key by routing through of maximum $log(n)$ nodes.

2.3 Resource Discovery

Resources in IoT can be IoT data or IoT devices. These resources are registered in the network and can be discovered by the clients. The process of discovery is to get the access address (e.g. URI or IP and port addresses) of IoT data, IoT devices or a combination of them as a result of discovering (i.e. querying) the resources in the network. The search techniques can be functional (event-based, location-based, time-related, content-based, spatio temporal-based, context-based, real-time and user interactive searching) or implementational (text-based, metadata-based or ontology-based approach) [25].

The resources can be registered in different parts of the network distributed among many nodes (Figure 1a) or in a centralised trusted entity (Figure 1b). The resource discovery [9] is a mechanism to return the access address of a resource based on the information provided during the lookup operation. The resource access address can be its IP and port addresses, its URI or other metadata and further links about the resource. The discovery process starts by issuing a query including the attributes of the required resources to be discovered. An attribute of a resource can be any information that describes it, such as its location, its type, etc. The query is issued by a client and sent to the discovery system. The query that is received by the discovery system is then processed and divided into sub-queries. As instance, the query can be divided based on the attributes of the required resources, and a sub-query is issued in the system for each attribute. The discovery system then finds and communicates with the responsible nodes to get the required information about the resources. After getting the list of resources, they are ranked based on some scoring methods and the final result is sent back to the requested client.

The data that are generated by the discovered resources in the system can be collected in either *request/response* or *publish/subscribe* patterns. In request/response, the data from the discovered resource is returned back to the clients based on their requests. As instance, Constrained Application Protocol (CoAP) [4] is a document transfer protocol that works based on a request/response approach on a client-server architecture. In the publish/subscribe, the discovered resource publishes its data to the clients that are already subscribed. The process can be done through a publish/subscribe server that is the middleware between the subscribed clients and the published resources. MQ Telemetry Transport (MQTT) [11] is a protocol that is based on publish/subscribe approach and facilitates one to many communications through a common node (i.e. broker). Resources publish the messages by sending them to the broker and on the other hand clients subscribe for a specific message in the broker.



(a) Decentralised Scheme (b) Centralized Scheme

Figure 1: Discovery and Access Mechanism

3 Related Work

Some researchers adopt the use of a centralised entity as part of their proposed models that manages some parts or all parts of the system. Cheshire and Krochmal [5] proposed a Domain Name System (DNS) based discovery for the IoT network. It defines a model on how the users register their resources and discover the resources based on the DNS protocol. The proposed model does not modify the underlying DNS protocol messages and codes and as a result is simple to implement. In this model, a centralized authority stores the registered resources and there is no additional security consideration to the original DNS protocol itself. Authors in [12] have proposed a large scale resource discovery to discover the devices and sensors in the IoT network by building a scalable architecture called Digcovery. The framework enables the users to register their resources into a shared infrastructure and to access/discover accessible resources by a mobile phone. Their proposed work is focuses on the discoverability of devices based on context-awareness and geo-location. Digcovery allows high scalability for the discovery based on a flexible architecture. However, it relies on a centralized point called *digcoverycore* for management and discovery.

Datta and Bonnet [8] proposed a resource discovery framework for IoT. The proposed framework includes a centralized registry that registers and indexes the attributes of resources. The attributes of the resources are used as the parameters during the discovery process through the search engine, that returns the access addresses of the discovered resources. The authors in [13] proposed a discovery model for IoT that performs the discovery based on various constraint parameters such as input/output (IO), precondition/effect and quality of experience (QoE). In the proposed model, a centralized directory server is used to register and discover the services in the IoT network. The discovery is done using semantic service description method OWL-S^{iot} that describes both the IoT services and discovery requests. Using the centralized scheme helps organizing the resources in an entity

that has a high computation capability, however, this centralized entity might turn into a single point of failure, which, if fails, the overall system will stop. This profoundly affect the availability and reliability of the system. Additionally, the centralized entity could turn into a bottleneck for the system affecting the overall system performance.

Several researches utilized the P2P scheme in the IoT network as a method for distributed resource discovery. The model in [15] removes the centralized entity by managing the fog nodes in a P2P scheme. It divides the resources into public resources that are discoverable by all clients and private resources that can be discovered by a subset of clients. In addition it provides other features such as multi-attribute discovery. However, the main drawback of this model is that by not considering the physical location of the resources and fog nodes it fails to keep the registration process low by using only the fog nodes that are in the same region of the registered resource. A particular emphasis on the links and nodes locality presents a Mesh-DHT in [30] that implemented in IEEE 802.11 wireless mesh network (WMN). The authors employed the Mesh-DHT for building a scalable DHT in WMNs. This approach enables an entirely distributed organization of information by building a stable, location-knowledgeable overly network. Because nodes primarily talk to physically nearby nodes, it allows minimizing the overhead in DHT communication of WMNs, therefore requiring fewer transmissions. However, the model can not reflect the locality of mesh routers in the overlay construction and therefore does not able to represent the locality of keys. Wirtz et al. [31] have been proposed an improved version of the DHT based service registration and discovery. Their work is based on their previous model (Mesh-DHT) and focuses to address its main drawback. Their proposed model partitions the global DHT overlay into different scopes with different degrees of locality that are hierarchically organized in levels. By choosing an appropriate level, a lookup can be restricted to only consider the locally available information. The put(key, value) and get(key) in DHT are extended to put(key, value, level) and get(key, level), respectively.

The authors in [21] proposed a single-gateway based hierarchical DHT solution (SG-HDHT) for an efficient resource discovery in Grids (i.e. Virtual Organizations (VO)). The model forms a tree of structured overlay and consists of a two-level hierarchical overlay network. It defines a global DHT and number of second level DHTs, a DHT overlay for each VO. Only one peer (called a gateway or super peer) in a DHT overlay of a VO is attached to the global level DHT of the hierarchy. The proposed resource discovery in this model deals with two different classes of peers: super peers and simple nodes. The lookup is directed to the super peer of the VO and then through the global DHT to the superpeer of the requested resource.

The authors in [1] proposed a wireless communication and computation framework that sustains the scalability for a massive increase of IoT devices. The researchers adopt the fog computing paradigm and therefore their proposed model enlarges the cloud-based solution by providing computing services close to the source of data generation. WMN nodes are used as the fog nodes in the proposed model. The authors have employed Chord [29], to generate a DHT based P2P overlay of fog nodes for resource discovery. This proposed model specifically targets the

underutilized processing power of devices for computing purposes. The discovery in this model is done by involving the fog nodes as brokers for the discovery of the required resources. Pahl and Liebald [23] introduced a distributed modular directory of service properties and a query federation mechanism based on virtual state layer (VSL) [24] that allows mapping complex semantic queries on the simple search. The presented modularaization adds little latency which makes it suitable for time-critical operations. The proposed model supports multi attribute discovery and allows adding new attributes in the system at runtime that fits the nature of the dynamically varying IoT.

Authors in [6] proposed an architecture consists of two discovery levels, local and global service discovery. It uses the P2P scheme for resource discovery, and IoT gateways are the peers in the P2P overlay. This architecture uses two layers: the Distributed Location Service (DLS) and the Distributed Geographic Table (DGT) [26]. The DLS is a DHT based architecture that works as a name resolution service by providing any required information to access any resource in the network, depending on its URL. The DGT builds a layer to distribute the information depending on the location of nodes, which can be used to discover the resources based on their geographic location information. The model successfully manages the registration and discovery based on the locations of the resources. Although DGT keeps the location data of the IoT gateways, but since DGT and DLS are loosely coupled then in order to discover the resources in a given location the system has to retrieve the IoT gateways data from DGT overlay and then lookup the DLS overlay for the required resources. In addition to keep the data as close as possible to the registered resources, Lamred aims to utilize a single DHT overlay and add no extra local overhead and low global overhead comparing to traditional DHT overlays. Furthermore, it aims to allow the participating nodes to join Lamred without using specific location based devices.

4 Location-Aware and Privacy Preserving Multi-Layer Resource Discovery (Lamred)

The resource discovery in fog/edge computing has some requirements that have to be addressed. Due to the distributed nature of the IoT gateways and the limited computing power of the IoT resources, the resource discovery model has to depend only on low computation processes and does not involve any centralized entity. Lamred allows four levels of discovery: *local discovery* that is limited to a single IoT gateway, *intra-regional discovery* that is limited to a local region (i.e. sub-region of a region set), *regional discovery* that is done in a specific geographical area and *public discovery* (i.e. location independent) that is done among all publicly registered resources, regardless of their locations. In addition, Lamred distinguishes between two types of resources, *public resources* that can be discovered by any client in the system (e.g. a public temperature sensor or a resource offering a public service) and *private resources* that can be discovered by a predefined subset of clients (e.g. private resources in a smart home or a local printer in an organization).

There are three main disjoint sets in Lamred: set of clients ($\mathcal{C}$), set of objects ($\mathcal{O}$) and set of gateways ($\mathcal{W}$). The finite set $\mathcal{C}$ consists of the IoT clients in the network. An object $o \in \mathcal{O}$ is any device in the IoT network with proper computational power that handles a resource u. Subsets of $\mathcal{C}$ and $\mathcal{O}$ are connected to different IoT gateways in $\mathcal{W}$. A gateway w's responsibility may vary from handling a few nodes (e.g. smart home) to hundreds of nodes (e.g. environmental monitoring). The proposed model creates a region-based DHT (RDHT) [17] overlay that provides a structured P2P method of addressing and discovery of the peers. The members of $\mathcal{W}$ (i.e. IoT gateways) represent the peers in the P2P overlay. Let $H(.)$ be a collision resistant one-way hash function with d bits message digest, $Enc_k(m)$ be an encryption of the message m using symmetric key k and $Sign_w(m)$ be a digital signature for message m generated by $w \in \mathcal{W}$ gateway.

4.1 Lamred Properties

The Lamred has been designed to address the requirements for resource registration and discovery in the IoT network. Table 1 shows a comparison of some of the supported properties in the different resource discovery models. In general, Lamred has the following properties:

- **Location Aware:** Lamred utilizes RDHT [17] that creates an overlay of IoT gateways divided logically into multiple region sets and local regions (i.e. sub-regions) in DHT overlay based on their physical locations. It generates a single overlay that can be generated without specific organizing entity or location based devices.
- **Multi-attributes:** Each resource has number of attributes. These attributes can be its location, its type, its provided service and so on. To discover a resource or a set of resources, in addition to their exact identifiers more than one attribute might be needed to get the precise result of the required resources. Lamred supports the multi-attributes discovery and the clients are able to discover the resources based on multiple attributes. In addition to the predefined set of attributes, participants in Lamred are able to create new attributes in real time.
- **Scalability:** The scalability describes the ability of the a decentralized resource discovery to adjust the registration and discovery process as the system grows in term of number of nodes. Lamred distributes the responsibility among many nodes that can continue working efficiently as the number of nodes grows.
- **Management:** Lamred provides defined interfaces for the authorized IoT entities to be able to add, remove, update and discover resources in the network.
- **Discoverability:** Because of the use of RDHT overlay, Distributed Address Table (DAT) [16] can be integrated as a part of Lamred (in a specific region

Table 1: Supported properties in Resource Discovery Models

Features	Decentralized	Location aware overlay	Multi attributes	Security considerations
Jara et. al. [12]	✗	-	✓	✗
Cheshire et. al. [5]	✗	-	✓	✓
Datta and Bonnet [8]	✗	-	✓	✗
Jia et. al. [13]	✗	-	✓	✗
Cirani et. al.[6]	✓	✓	✗	✗
Wirtz et. al. [30]	✓	✓	✗	✗
Wirtz et. al. [31]	✓	✓	✗	✗
Mokadem et. al. [21]	✓	✗	✗	✗
Shabir et. al. [1]	✓	✗	✓	✗
Kamel et. al. [15]	✓	✗	✓	✓
Pahl et. al. [23]	✓	✗	✓	✓
Lamred	✓	✓	✓	✓

in RDHT) to allow discoverability of all resources in the network including as instance those behind the Network Address Translator (NAT).

- **Responsibility Definition:** Each node in RDHT overlay of Lamred is aware of the range of its responsibility to registering a subset of resources. This results that the clients that need to discover a resource in the system being aware of the specific IoT gateway in Lamred that is responsible to store the required information to access that specific resource, and issue a discovery request to that specific node.
- **Discoverability Range:** Lamred uses a private/public architecture and is able to keep some of the resources private and only discoverable by the authorized clients in the IoT network.

4.2 Security model

An object $o \in \mathcal{O}$ that needs to register its private resource in Lamred has a predefined set $\mathcal{F}_o \subset \mathcal{C}$ of friend clients. The members of $\mathcal{F}_o$ are able to discover the privately registered resource of object o. We assume that from the viewpoint of any object $o \in \mathcal{O}$ in Lamred, the set of friends $\mathcal{F}_o$ are *honest* nodes. The rest of the clients $\mathcal{R}_o = \{r \in \mathcal{C} \setminus \mathcal{F}_o\}$ can be assumed to be *malicious*. In the case of the finite set of IoT gateways $\mathcal{W}$ in Lamred, we have to assume that there is no cut containing malicious nodes only in the communication graph composed of the clients, objects and gateways (otherwise, the malicious nodes together could make the communication impossible). More precisely, we assume that for a given resource u of an object o for every friend $f \in \mathcal{F}_o$ there exist a path $(w_1, w_2, \ldots, w_k)$ in the communication graph such that the object o is connected to w_1, the friend client f is connected to w_k and all of $w_1, \ldots, w_k$ are *semi-honest*. The semi-honest entities

are assumed to follow the protocol properly, but they might store the received data locally in an attempt to get more information from the stored data.

Beside these properties, the nature of the communication model also regulates the applicable security. In Lamred only the IoT gateways assumed to be able to use public key cryptography, while the objects handling the IoT resources can encrypt and decrypt messages using symmetric keys only because of their limited computation power. In case of IoT gateways, we suppose that every $w \in \mathcal{W}$ can generate a digital signature $Sign_w(p)$ of any transmitted packet p. The proposed construction is supposed to achieve security requirements in the computational sense, i.e. we assume PPT adversaries (definition 2) with negligible success probabilities when attempt to attack the scheme. A function has a negligible success probability if the success occurs with a probability smaller than any polynomial fraction if the size of the input exceeds a given bound [2]. The private resources are registered and discovered by an added private tuple (see Section 4.5). The goal of the security model in the privately registered resources of Lamred is to allow friend clients to securely and anonymously discover the private resources, which comes from the following security properties:

- **Resource anonymity**: Every PPT adversary can learn any connection between a given private tuple and a given private resource with negligible probability only.
- **Resource privacy**: Every PPT adversary can learn the address of a resource from a given private tuple with negligible probability only.
- **Unforgeability**: Every PPT adversary is able to generate, remove or update a valid private tuple of a resource on behalf of a given honest object with negligible probability only.

4.3 Location Regions in Lamred

Lamred consists of maximum 2^g regions with maximum of 2^d IoT gateways in each region. The regions are grouped in $2^{g/2}$ sets, each set with one representative region and $2^{g/2} - 1$ local regions. Consequently, an identifier of a node in Lamred consists of three concatenated parts: region set id, local region id and local node id and is $(g + d)$-bit long. During the creation of identifiers in Lamred, two hash functions are used. The first hash function generates g-bit output digest based on a given information of the region set and the local region, while the second hash function generates d-bit output digest based on the given node information. g and d parameters can have same value and the same hash function can be used to generate the different parts of the identifiers. There are two specific generic regions in Lamred, namely *private* and *public* regions. Each node joins private and public regions regardless of its physical location. In addition to that, a new node joins a local region in the Lamred based on its location. Figure 2 illustrates the regions in RDHT overlay of Lamred.

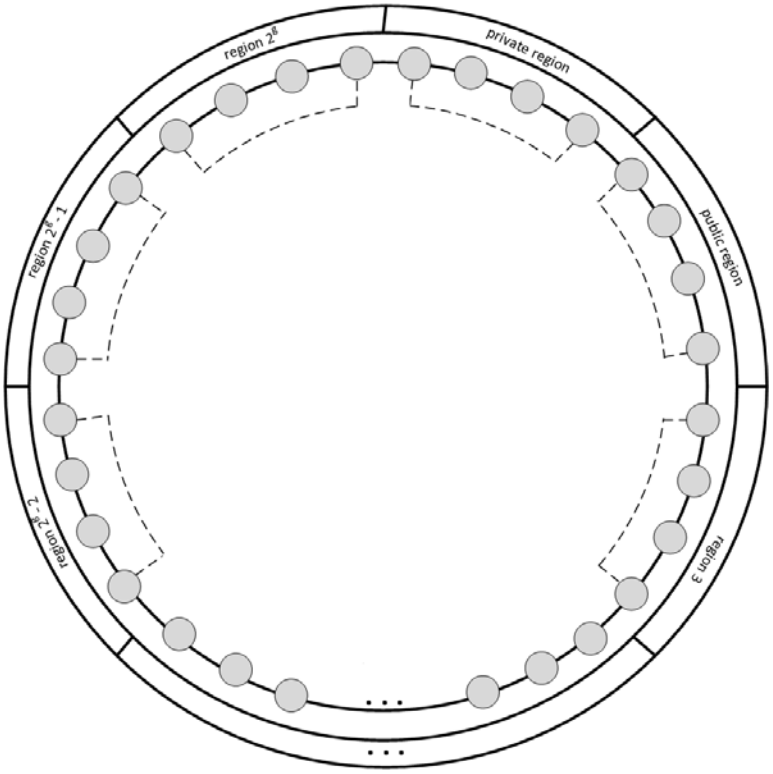


Figure 2: Regions in RDHT

The regions are generated by feeding the information of the locations to the hash function that outputs a g-bit digest. The given input information of the locations can be represented by human readable names of regions or a specific prefix of latitude/longitude data. Each region set has a representative region, in which the nodes in the that region represent other nodes in the region set. To create a region id, the information of the representative region of that region set is fed to the hash function and the first left $g/2$ bits represents the first $g/2$ bits of the generated region id. The local region information is then fed to the hash function and the last $g/2$ bits is taken that will represent the second $g/2$ bits of the generated region id. The representative region itself, will have the last $g/2$ bits all set to zero. As a result, all the regions in each of the $g/2$ region sets in RDHT share the same $g/2$ prefix bits. Because of the Avalanche effect property [32] of the hash function algorithms, each subset of a generated digest by the hash function should be affected equally as any other subset of the digest. Therefore, generating the region id by taking $g/2$ bits from the g bits digest of representative region and $g/2$ bits from the g bits digest of the local region should not affect the randomness of the generated identifier. The remaining d-bit of the identifier of a node is generated

by hashing the information of the IoT gateway (e.g. its IP address). Figure 3 shows the generation of the identifier of a node in Lamred.

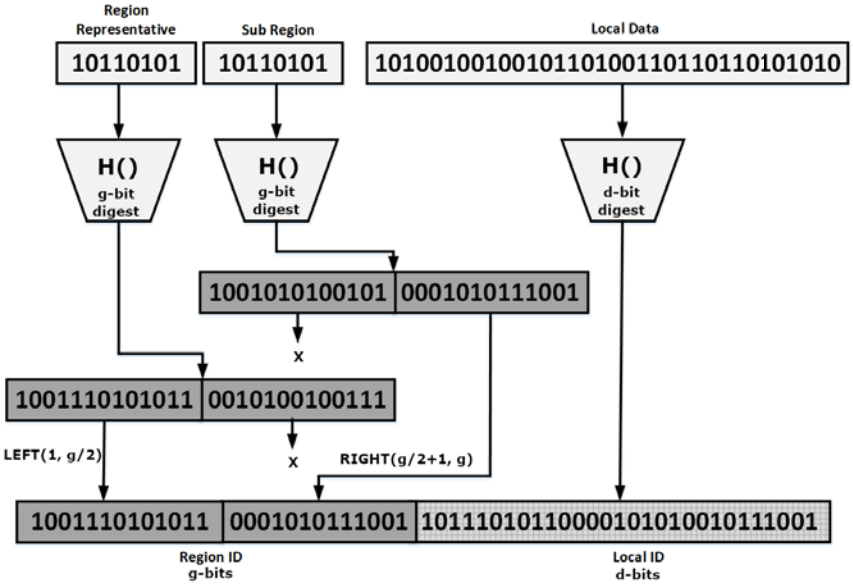


Figure 3: Identifier generation in Lamred

A new IoT gateway $w \in \mathcal{W}_{rg} \subset \mathcal{W}$ joins Lamred by registering in its local region, along with other members of $\mathcal{W}_{rg}$ in the same local region. This is done by hashing the location information of the representative region and its local region to get the first g bits of the identifier of node w and then hashing its unique information (e.g. IP address of w) resulted in the rest d bits of the identifier of node w. In addition to the local region, the newly joined node w can join private and public regions as well. This is done by first hashing its unique information (e.g. IP address of w) to be able to generate two identifiers that are used in private and public regions. The generated identifier in the private region starts with g zeros followed by the d bits output of the hash function used by w. The generated identifier in the public region starts with $g-1$ zeros followed by a single bit 1 and the d bits output of the hash function used by w. The joining process is done through an *introducer node* that is already a member of Lamred. The joining process of a newly joined node w starts by sending a look up request through the *introducer node* for its own identifier (i.e. the newly generated identifiers of node w) in both private and public regions, as well as in its own local region.

Similar to Kademlia [20] there are two general α and k parameters in Lamred that determine the parallelism and system wide replication, respectively. Each node in Lamred, has d lists of the k-buckets [20] that includes the access addresses to the nodes in the same region. In addition, each node in any region of a region set should keep $g/2$ lists of the k-buckets that includes access addresses to all representative

regions in all region sets in RDHT, including the public and private regions. The nodes in the representative region of any subset, keep $g/2$ lists of the k-buckets that includes access addresses to all local regions in the region set. As a result, the nodes in the representative regions have $d + g$ lists and all other nodes in any region in RDHT have $d + g/2$ lists. Each of those lists has maximum of k entries. Considering the size of the DNS domain cache in Raspberry Pi that is defaulting to 10,000 entries[1], storing the access addresses of maximum k IoT gateways in maximum $d + g$ lists does not require any additional storage consideration in the Lamred peers. Figure 4 shows an example of a RDHT overlay of Lamred with 3 region sets, in addition to the public and private regions. In this tiny example the hash function of both region and local identifiers generates a 4-bit digest, therefore, the identifier of each node consists of 8-bits that includes 4-bits region identifier and 4-bits local identifier. As instance, initiator peer with identifier 10001110 wants to get the access address of a resource that is stored in the destination peer 11101111. Since the destination peer is in region id (1110), the representative region that this peer belongs to is (1100). Therefore, the initiator peer sends the request to the node (11001110) in the representative region which is then directed to the specific region and finally to the destination peer in the required region.

4.4 Public Resource Registration and Discovery

A resource u in the network has its specific access address and a set of attributes that describe its properties (e.g. its type, its provided service, etc.). When an object $o \in \mathcal{O}$ wants to register its resource u in the network, it has to add the required information in Lamred through a member of $\mathcal{W}$ that is directly connected to. This set of information includes the tag that is generated by hashing the attribute type of the resource, the value of the added attribute, the ownership information and the access address to resource u as illustrated in Figure 5. After adding the tuple $(tag, value, ownership, data)$ of resource u to Lamred, it can be discovered by all clients in the network. There are two options for registering a resource in the network. The first option which is the default choice for a resource is to register it in the local region, i.e. in the same region that it belongs to. Since registering it locally ensures that the tuple will be stored in a node in the same geographical region, it requires less time and overhead for registration. The resources that do not depend on a specific location or provide services that are location-independent, can register themselves in the public region as well. In addition to that, the directly connected IoT gateway (i.e. the IoT gateway w that the object o is connected to) keeps a copy of the registered resource locally in the cache for a specific time depending on the caching expiry parameters.

The overall workflow of resource registration and discovery is shown in Figure 6. An object $o \in \mathcal{O}$ registers its resource u in the network as tuples of $(tag, value, ownership, data)$. The set of the attributes that describe resource u are fed to the hash function to generate the *tag* parameter. The *value* in the added

[1] `https://docs.pi-hole.net/ftldns/dns-cache/`

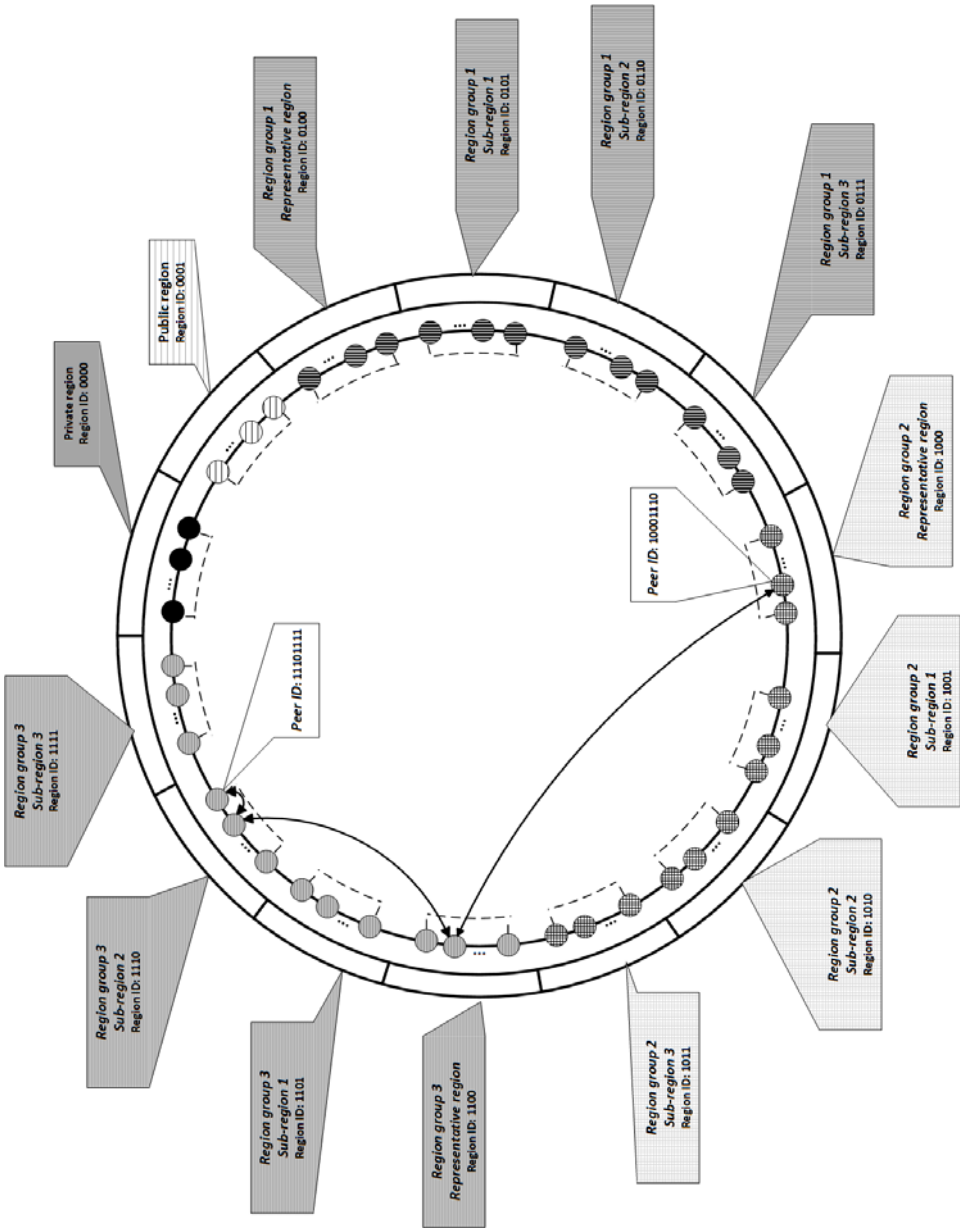


Figure 4: An example of Lamred implementation with 4 bits per region id and local id

tuple indicates the actual value of each of the attributes of the registered resource. During resource registration, the object o generates a random number r and adds

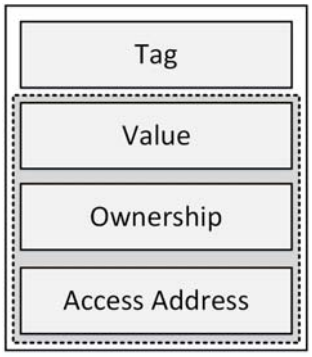


Figure 5: The public tuple structure in Lamred

its hashed value as a later proof of *ownership* of the generated tuple. Revealing the pre-image of the hash value (i.e. r) guarantees the ability of proving the ownership of the tuple that is used during updating or removing it from the Lamred. The access address (i.e. *data* in the tuple) parameter of the resource u might consist of its address, URI or other metadata about the resource u. The tuple is then stored in RDHT based on the its tags with a predefined number of replicas. The actual number depends on the *replication factor* rp. Choose an appropriate rp parameter depends on the nature of the network. As a general rule for choosing the appropriate rp value, the probability of existence of a subset of offline nodes in Lamred $\mathit{Offline} \subset \mathcal{W}$ with cardinality greater than the number of replicas has to be negligible ϵ. This is shown in equation 1. In addition, the existence of replicas increases the system performance by reducing the access load on any specific node in Lamred.

$$P(\|\mathit{Offline}\| \geq rp) < \epsilon \tag{1}$$

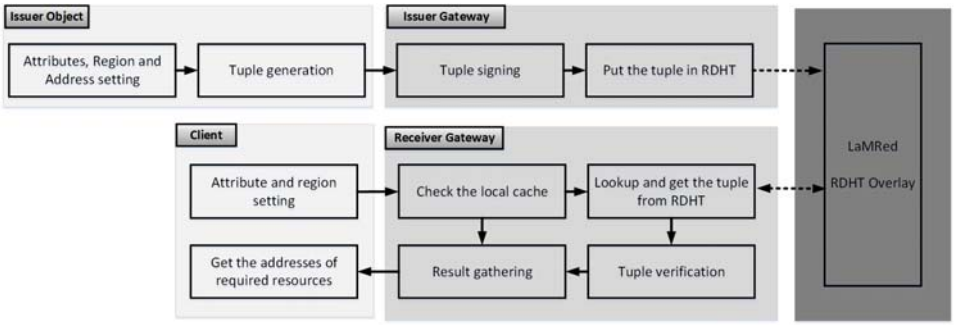


Figure 6: Overall workflow of resource registration and discovery in Lamred

Let $\mathcal{W}_{rg} \subset \mathcal{W}$ be a subset of IoT gateways in a specific region that the resource u has been registered in. In addition to storing the tuples locally in the directly connected IoT gateway $w \in \mathcal{W}_{rg} \subset \mathcal{W}$ and depending on the replication factor, the *close nodes* in the same local region $\mathcal{W}_{rg}$ to tag parameter are responsible for storing $(tag, value, ownership, data)$ tuple. If a node with identifiers id_w and a tuple with tag tag_v are close or equal, then we denote it with $id_w \approx tag_v$. The model does not depend on any specific *distance function (dst)* to compute the closeness. It can be any particular distance function. Metrics such as bitwise exclusive or (xor) [20] can be used to compute dst value.

Let $\mathcal{I}_d$ be a set of all possible sequences of d-bit binary digit (i.e. identifiers) in region rg and each peer $w \in \mathcal{W}_{rg} \subset \mathcal{W}$ has an identifier $id_w \in \mathcal{I}_d$ and each resource u has a $tag_u \in \mathcal{I}_d$. Let define the following set of peers

$$M(u) = \{w : tag_u \approx id_w, \nexists w' \mid dst(id_{w'}, tag_u) < dst(id_w, tag_u)\} \tag{2}$$

The set $M(u)$ links each resource u depending on its added attribute tag_u to a node $w \in \mathcal{W}_{rg}$ that its identifier $id_w \in \mathcal{I}_d$ is close or equal to tag_u. The cardinality of $M(u)$ depends on the replication factor rp parameter. The procedure of registering a public resource u in the network consists of three steps:

- `Tuple Definition and Generation`: The object o and based on the attributes that describe the resource u generates the tags, i.e. hash value of the attributes. Each of the tags is put along with their values, the ownership parameter that is the hash value of a randomly generated number r and the access address to the resource u and send the generated tuples to directly connected w. In addition to that, the object o determines whether u has to be stored in the same region or in the public region.

- `Tuple Signing`: In this step the appropriate set of tuples of the resource u are signed by $w \in \mathcal{W}$.

- `Resource Registration`: The gateway w registers the resource u by storing the tuples at the corresponding nodes in Lamred.

The public resources that are registered without any restrictions in Lamred can be discovered by all clients in the network based on their attributes and the registered regions. Lamred allows discovery of the registered resources based on one or more attributes. A client $c \in \mathcal{C}$ lookup for a resource by sending a lookup request with the required set of attributes, their values and the required region to the node w in Lamred that is directly connected to. The node w after receiving a discovery request from a client c generates the appropriate tags for the discovery process based on the received attributes. The discovery process contains three main steps as follows.

- `Query Generation`: In the first step, the node w generates the set of tags based on the received attributes from a client c. This is done by hashing each

of the requested attributes in the client's request. In addition to that, the region id is also added to the generated tag.

- `Lookup`: The second step starts by issuing the lookup request by w in Lamred. The result $\mathcal{R}_i$ of each of the lookup operations is a set of data parameters that indicates the resulted resources based on the given attribute i and its required value.

- `Result Gathering`: After receiving the results and verifying them based on their attached digital signatures, the intersected members of sets $\mathcal{R}_0 \cap \mathcal{R}_1 \cap \cdot\cdot \cap \mathcal{R}_n$ will be returned as a result to the requested client c. In this step and prior to returning the result to client c, some scoring methods can be applied.

The tuples of the registered resources are remained in Lamred based on the caching expiry parameter. In addition to that, an object is able to update the data or remove its registered resource from Lamred by issuing a request including the pre-image of the ownership field in the added tuple. The request is signed by the directly connected node in Lamred, w and is sent to the corresponding node in Lamred. After checking the ownership of the tuple (i.e. $H(r) = ownership$), the requested tuple is updated by a new tuple or removed from Lamred based on the received request.

4.5 Private Resource Registration and Discovery

Every object o in the IoT network is able to keep a resource private and discoverable only by a predefined set $\mathcal{F}_o$ by generating a private tuple as illustrated in Figure 7. An object o has a set of its friends $\mathcal{F}_o = \{f_1,...,f_n\} \subset C$ that can be communicated with in a secure and trusted way. The members of a friend set $\mathcal{F}_o$ of an object o are connected through members of $\mathcal{W}$ but they are not part of RDHT overlay itself. Each private resource has a private identifier id_u that is chosen uniformly at random from a given range, e.g. from bit strings of length 512. The identifier id_u is known only by the members of $\mathcal{F}_o$. Additionally, each $c \in \mathcal{C}$ has also a private identifier id_c that is chosen uniformly at random from a given range. The object o stores the private identifiers of each $f \in \mathcal{F}_o \subset \mathcal{C}$ locally. In addition, for every object o and for each $f \in \mathcal{F}_o$, an initial value (IV_{of}) and a common secret key (k_{of}) are generated and shared between them on a secure channel. The key k_{of} is used to encrypt the indirect transmitted data between them. These keys are stored at each node locally at the setup phase and its future distribution scheme is out of the scope of this paper.

If an object o registers its resource u privately, only the members of $\mathcal{F}_o$ can discover and access this specific resource of o. To do so, an object o has to generate a $privateTag_{uf}$ for a resource u and every friend client $f \in \mathcal{F}_o$ using equation 3. The access address of the private resource is then encrypted using the shared key k_{of}. A random number r is also generated and its hashed value is added as a later proof of *ownership* of the generated private tuple. A private resource can be registered privately in the local region or in the private region. While

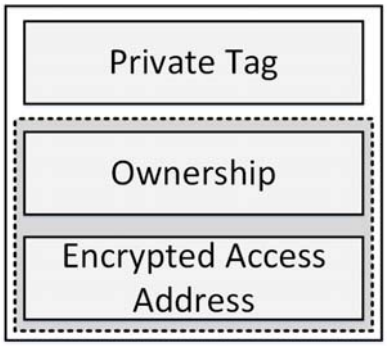


Figure 7: The private tuple structure in Lamred

registering a private resource in the private region does not guarantee the low latency process, but it hides the actual region that the private resource belongs to. On the other hand, registering a private resource in the local region ensures a low latency process, but reveals its local region. The decision of whether the private resource should be registered in the same local region or in the generic private region is made by the object o that handles the private resource. After generating the tuples as $(privateTag_{uf}, ownership, encryptedaccessAddress)$, the corresponding node in Lamred receives the resulted tuples (a single tuple for each $f \in \mathcal{F}_o$) from the directly connected object and put them in the same local region or the private region of RDHT. After registration, the members of $\mathcal{F}_o$ can discover the registered private resource by computing its private tag. These private tags are not permanent and used only once. The $privateTag_{uf}(new)$ parameter can be calculated using $privateTag_{uf}(old)$, id_u and id_f values. At any given time, the current private tag of a resource is computed as 3:

$$privateTag_{uf}(new) = H(privateTag_{uf}(old) \oplus id_u \oplus id_f) \tag{3}$$

where $privateTag_{uf}(old)$ is the previous private tag of the resource u (i.e. the output of the previous hash) and the initial value is $privateTag_{uf}(old) = H(IV_{of} \oplus id_u \oplus id_f)$. The one-time private tag ensures that the IoT gateway $w \in \mathcal{W}$ that $id_w \approx privateTag_{uf}$ is not the same during the life cycle of the resource. Similar to publicly registered resources, the private tuple of a privately registered resource can be updated or removed from Lamred by issuing a request and revealing the pre-image of the ownership field in the added private tuple. Although the discovery process of a private resource in the network resembles the public discovery, but there are two differences. First is that in order to be able to discover a resource u that is handled by o, a client has to be able to compute its private tag, i.e. being a valid member of $\mathcal{F}_o$. Secondly, after receiving the discovery result, the returned access data is confidential and can be read only by knowing the secret key k corresponding to this specific node.

5 Evaluation

5.1 Security Analysis

Theorem 1. *If $H(.)$ is a one-way hash function then the system satisfies resource anonymity.*

Proof. Suppose that the private tuple

$$P = (P_1|P_2|P_3) = (H(privateTag_{uf}(old) \oplus id_u \oplus id_f)|H(r)|Enc_{k_{of}}(accessAddress))$$

is stored in Lamred by object o to register the resource u that can be discovered only by its friend client $f \in \mathcal{F}_o$. Note that, only P_1 includes some information related to the private resource u (i.e. id_u), hence we can deal with this part of the private tuple only, hence the goal of the adversary is to compute id_u from the tuple. Let assume that the adversary knows $privateTag_{uf}(old)$ also (e.g. from previous communications). First suppose that a client $m \in \mathcal{C} \setminus \mathcal{F}_o$ wants to learn some information. Additionally, we can suppose that $m \in \mathcal{F}_f$, i.e. m knows id_f. If m could find a pre-image of $H(privateTag_{uf}(old) \oplus id_u \oplus id_f)$, and if she could remove $privateTag_{uf}(old) \oplus id_f$ then she can compute id_u. However, since $H()$ is a one-way function, m can find any x with $H(x) = P_1$ with negligible probability only. The remaining nodes in Lamred outside $\mathcal{F}_f$ are in a much hopeless situation, since even if they are assumed to find a pre-image of the hash, after that they have to remove $privateTag_{uf}(old)$ and id_f and the later is chosen randomly arising unconditional resource anonymity in this case. This completes the proof. □

Theorem 2. *If Enc is a computationally secure encryption then the system satisfies resource privacy.*

Proof. Suppose that the private tuple

$$P = (P_1|P_2|P_3) = (H(privateTag_{uf}(old) \oplus id_u \oplus id_f)|H(r)|Enc_{k_{of}}(accessAddress))$$

is stored in Lamred by object o to register the resource u that can be discovered only by its friend client $f \in \mathcal{F}_o$ and a malicious node $m \in \mathcal{W} \cup (\mathcal{C} \setminus \mathcal{F}_o)$ wants to discover and learn the access address of the registered private resource. Note that, only P_3 depends on the access address of the private resource, hence we can deal with this part of the private tuple only. This last part of the tuple is the $accessAddress$ encrypted with a computationally secure encryption. Therefore, without the knowledge of the symmetric key k_{of} the address $accessAddress$ can be computed with negligible probability only. This completes the proof. □

Theorem 3. *If $H(.)$ is a collision-resistant one-way hash function and Enc is a computationally secure encryption, then the system satisfies unforgeability.*

Proof. Suppose that the object o registers the resource u and the actual private tuple

$$P = (P_1|P_2|P_3) = (H(privateTag_{uf}(old) \oplus id_u \oplus id_f)|H(r)|Enc_{k_{of}}(accessAddress))$$

is stored in Lamred. Let $m \in \mathcal{W} \cup (\mathcal{C} \setminus \mathcal{F}_o)$ be a malicious node and first suppose that m wants to remove or update this tuple. Then m has to compute the pre-image of P_2, which is possible with negligible probability only since $H(.)$ is one-way.

Next, suppose that m wants to generate a new valid private tuple. To achieve this, the malicious node m has to first compute the new private tag (i.e. P_1) and then replace the part containing information related to $accessAddress$ (i.e. P_3). We will show that neither part of the tuple can be computed with non-negligible probability. First suppose that m wants to compute $P_1' = H(P_1 \oplus id_u \oplus id_f)$, furthermore assume that $m \in \mathcal{F}_f$ (i.e. m knows id_f) and $privateTag_{uf}(old)$ is also known by m. Then the only remaining part necessary for P_1' is id_u and only P_1 and $privateTag_{uf}(old)$ depends on this identifier. In both cases m has to compute the pre-image of the hash function $H(.)$ which can be done with negligible probability only, since $H(.)$ is a one-way function. Finally, suppose that m wants to compute $P_3' = Enc_{k_{of}}(accessAddress')$ for a fake address $accessAddress'$. Such fake address can be found with negligible probability since Enc is a computationally secure encryption. This completes the proof. □

5.2 Performance Analysis

In addition to proving the security properties in Lamred, the main concern is to keep the data of the registered public and private resources in the system as close as possible to the point of origin to prevent the high latency of long distances. In order to study the performance of Lamred and validate its feasibility and reliability, several issues such as region sizes, required preparation time for registration and discovery in constrained IoT devices, local and global discovery, and the affect of local cache size and churn on Lamred have been investigated. The network latency has been taken into consideration for measuring the performance of Lamred. Table 2 shows the assumed random parameters of real-time latency[2] for each of the different network links in the system.

Table 2: Network parameters

type	parameter
local connection latency	2 ms
sub-regional latency (local region)	3 - 8 ms
intra-regional latency (region set)	10 - 30 ms
long distance latency	80 - 120 ms

[2] `https://wondernetwork.com/pings`

We should call the reader's attention to the fact that the IoT gateways are the peers in RDHT and not the IoT clients or IoT resources. The members of $\mathcal{C}$ and $\mathcal{O}$ (i.e. clients and objects that handle IoT resources) are not part of RDHT itself and are connected through the peers in RDHT. The Kademlia implementation[3] of PeerSim simulator[22] has been used for the performance experiments. The implementation has been modified to fit our proposed model. In the implementation and as with uTorrent[4], the popular implementation of Kademlia, system wide replication is set to 8 and the lookup parallelism is set to 4. The results of researches [14][27] that focus on studying these two factors and other parameters in Kademlia [20] implementation to improve the lookup latency in DHT based implementation can be applied on Lamred. The system performance has been tested using a simulated Lamred network with 400 million to 2 billion IoT gateways. The IoT gateways are distributed and grouped in 200 region sets with 200 regions per region set (i.e. overall 40,000 regions with 10,000 to 50,000 IoT gateways per region). Table 3 shows the resource discovery latency in a local region. Figure 8 shows the resource discovery in Lamred with different region size and discovery scope. The sub-regional discovery is done within the same region, intra-regional discovery is done between two regions that are within the same region set and regional discovery (i.e. long distance discovery) is done between two regions that are in two different region sets. Due to huge number of IoT gateways in RDHT, the intra-regioal and regional discovery have been simulated in different stages. Without loss of generality, we assumed that no churn occurred and no cache has been used in Lamred during these tests.

Table 3: Resource discovery in a region in Lamred

region size	discovery latency
10,000	45.27 ms
20,000	47.14 ms
30,000	48.1 ms
40,000	48.9 ms
50,000	49.7 ms

To evaluate the efficiency of the Lamred for handling issues of robustness, availability, and replication, we performed a set of experiments where we introduced churn in the network. Over 100 to 2000 milliseconds intervals and for a period of 120 seconds, we randomly either killed an existing IoT gateway or started a new one in a region of 10,000 nodes. During the evaluation, resource discovery rate of 100 requests per second have been issued. As shown in the presented result in figure 9, there is 11 ms delay comparing to the network without churn in the discovery time in Lamred when the churn rate is 0.1 second (i.e. every 100 millisecond either an IoT gateway leaves or joins Lamred) and less than one millisecond delay when

[3] `http://peersim.sourceforge.net/`
[4] `https://www.utorrent.com/`

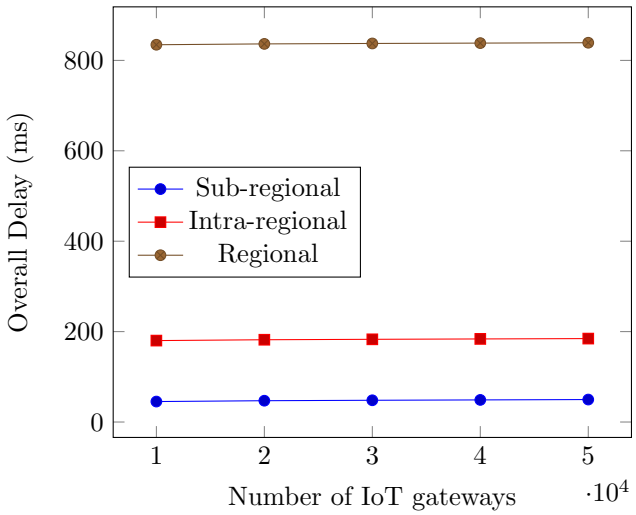


Figure 8: Regional Resource Registration Delay

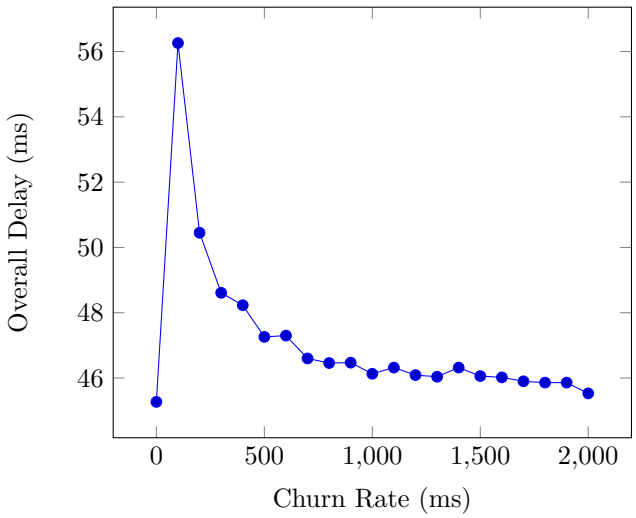


Figure 9: Churn affect on Lamred

the churn rate is higher than 1.6 second between each occurrence.

Figure 10 shows the affect of cache on Lamred. During the evaluation in a region of 10,000 IoT gateways and 1000 requests per second, the probability of discovering a resource that has been already resided in the local cache has been set to 0.05 - 0.25. The analysis showed that the local cache in Lamred nodes that includes the

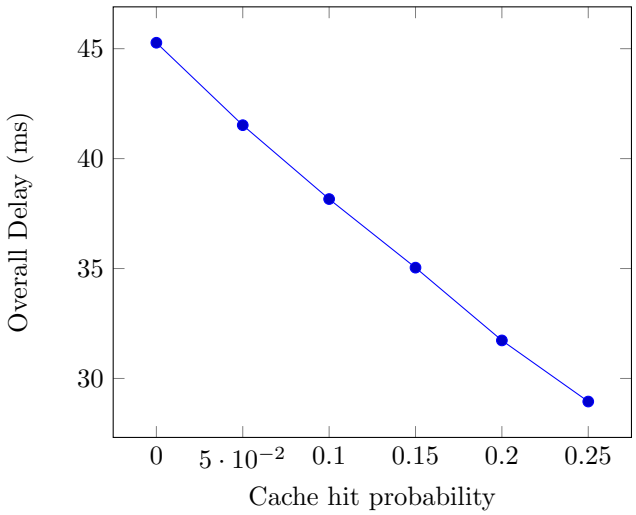


Figure 10: Cache affect on Lamred

locally registered and frequently discovered resources, improves the overall delay linearly.

As part of the evaluation, Lamred has been compared with the centralized service discovery (CSD) [13] and the decentralized resource discovery (DRD) [15] models. Since the DRD model [15] uses the IoT gateways without considering their locations, during analysis and comparison we simulated this model by creating a region set and assuming that the resources are registered in the regions without considering the locations of the resources. The direct matching scheme that has the minimum response time in the CSD model [13] has been used. There are 1000 IoT objects that have been distributed uniformly at random among a region in Lamred with 5,000 - 10,000 IoT gateways. As it appears from figure 11, although the resource discovery in centralized discovery is fixed, but the lookup process of discovering a resource in the network of a centralized scheme is higher than the proposed model. At the same time, as it is notable, when in the proposed model the number of gateways in the system increases the delay of the lookup process increases logarithmically. The reason is the use of the DHT overlay for discovery in which the lookup time among n peers is $log(n)$.

Lamred shows that it has a low latency that makes it suitable for IoT network with large number of IoT resources. The latency in a region of Lamred has been compared with the latency in some of the recent works and the result is listed in Table 4.

The private resource registration and discovery follows a different approach than other regions, as discussed in Section 4.5. In this case the object has to generate the private tag of the resource to be used in the private region of Lamred and on the other hand, the client application has to calculate the private tag in order to

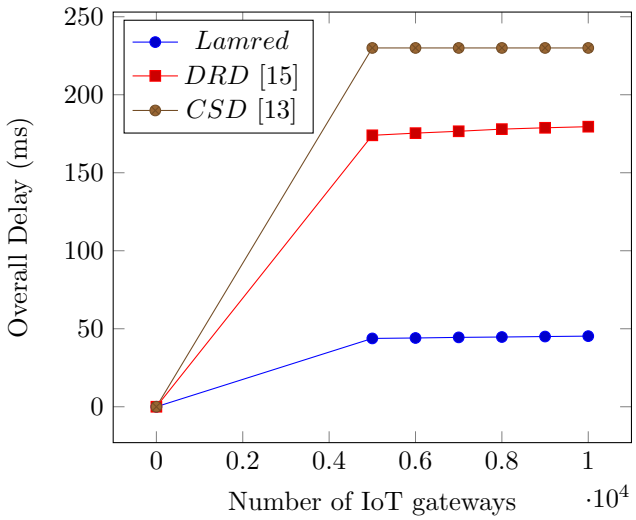


Figure 11: Resource discovery delay in different models

Table 4: Latency in resource discovery Models

Model	Properties	Latency
Datta and Bonnet [8]	Search Engine Based Resource Discovery	450-600 ms
Jia et. al. [13]	Centralized Resource Discovery	230 ms
Kamel et. al. [15]	Decentralized Resource Discovery	150 ms
Pahl et. al. [23]	4 predicates/ search providers	80 ms
Lamred	A region with 10,000 IoT gateways	45 ms

be able to discover the private resource and get the encrypted access address of it. The private tag generation in equation 3 has been tested on an MCU with single-core 32-bit 80 MHz microcontroller. The SHA256 [10] is used as hashing algorithm for tag generation and AES-128-CBC is used as encryption algorithm. For analysis and implementation of SHA256 and AES algorithms on the MCU, the Crypto library[5] for the ESP8266 IoT devices has been used. During the test, each of the cryptographic operations has been repeated 20 times, and their mean value is registered. The average time required to perform the encryption, decryption, hashing and the private tag generation and discovery are shown in Tables 5 and 6.

[5]`https://github.com/intrbiz/arduino-crypto`

Table 5: Required operation time by the microcontroller for private resource registration

Operation	**Required time**
XOR operation	8 ms
RNG	5 ms
SHA256 (Private Tag and RN)	2 * 227 ms
AES-128-CBC encryption	288 ms
Tag generation (Registration)	**805 ms**

Table 6: Required operation time by the microcontroller for private resource discovery

Operation	**Required time**
XOR operation	8 ms
SHA256 (Private Tag)	227 ms
AES-128-CBC decryption	348 ms
Tag generation (Discovery)	**583 ms**

5.3 Complexity Analysis

Suppose that Lamred consists of 2^g regions, divided into $N_{RegionSet}$ region sets. Let's suppose that the cardinality of IoT gateways in the private and public regions are N_P, and in the local regions $R1$, $R2$ and $R3$ of RDHT overlay are N_{R1}, N_{R2} and N_{R3}, respectively. Suppose that both $R1$ and $R2$ are in the same region set that includes N_{RS1} local regions, and $R3$ is in a different region set. We discuss the complexity of the proposed model in five cases:

- Sub-regional registering or discovering a resource in the same local region (RD_{local})
- Location independent registering or discovering a resource in the private/ public regions (RD_{pp})
- Discovering a resource that is stored in the cache of an IoT gateway (D_{cache})
- Intra-regional discovering a resource in the same region set (RD_{RS})
- Regional discovering of a resource in a different region set than the client region ($D_{regional}$)

Registering or discovering a resource in the same region $R1$ that the object and client belong to is done by the corresponding peer $w \in \mathcal{W}_{R1}$ and is equal to $RD_{local} = O(log(N_{R1}))$. Registering or discovering a resource in the private or public regions regardless of its location is done by first reaching a node in the target private or public regions and then finding the exact node in these regions

responsible for storing the access address of the required resource. Since each node in Lamred has the access addresses of nodes in public and private regions, it takes $O(1)$ to reach each of these two regions and then perform a lookup request for the exact required node. Therefore, registering or discovering a resource in the private or public regions depends on the number of nodes in each of these regions and is equal to $RD_{PP} = O(log(N_P))$.

Each IoT gateway in Lamred keeps a copy of the registered or previously discovered resources locally for a specific time depending on the caching expiry parameters. If a client requests to discover a resource that resides in the cache (which happens for the frequently discovered resources), then the result is returned directly to the client and is equal to $D_{cache} = O(1)$. If the client and the discovered resource are in regions $R1$ and $R2$ that are in the same region set including overall N_{RS1} local regions, the discovery access time takes $RD_{RS} = O(log(N_{RS1}N_{R2}))$ and is done in two stages. Firstly, it takes $O(log(N_{RS1}))$ to reach the target region (i.e. $R2$) and then it takes $O(log(N_{R2}))$ to discover the required resource by reaching the specific responsible node in target region $R2$. Discovering a resource in region $R2$ by a client belongs to region $R3$ that is in a different region set rakes $D_{regional} = O(log(N_{RegionSet}N_{RS1}N_{R2}))$ and is done in three stages. Firstly, accessing the representative region of the region set that the target region $R2$ belongs to takes $O(log(N_{RegionSet}))$ based on the number of available region sets in Lamred. Then, reaching the region $R2$ takes $O(log(N_{RS1}))$. Finally, it takes $O(log(N_{R2}))$ to perform a lookup and discover the required resource by reaching the specific responsible node in target region $R2$.

6 Conclusion

In this paper a location aware and privacy preserving multi layer model of resource discovery (Lamred) in IoT has been proposed. It adopts the peer to peer (P2P) scheme by utilizing Regional Distributed Hash Table (RDHT), a proposed version of DHT. Lamred ensures that there is no single point of failure in the system and the network can be easily scaled without any need of a reorganizing and synchronizing authority. The RDHT overlay is generated by taking into consideration the physical location of IoT gateways in the system. Resources are not part of RDHT overlay, but they can be registered locally, globally or privately different regions in RDHT through an IoT gateway. On the other hand, clients can discover the resources based on one or more attributes of the required resources. During the discovery phase, the client can choose a specific local region or the public region for the discovery of the resources. The private resources that are registered privately either in the local region or in the private region can only be discovered by a predefined set of clients in Lamred. During the evaluation, Lamred showed a lower latency comparing to the centralized and location-independent decentralized resource discovery models. In addition, the required security properties of the registered resources in Lamred, namely resource anonymity, privacy, and unforgeability have been proved.

Some open problems remain related to the proposed model. On one hand, while

the current model supports registering private resources in Lamred, but it offers a two-level binary policy and can not define the set of attributes of clients that are able to discover privately registered resources. This problem has to be addressed in future works. On the other hand, in Lamred a separate lookup in the DHT overlay for each of the attributes is issued which will add a significant overhead to the system, there should be a future study to improve the efficiency of the discovery process.

References

[1] Ali, Shabir, Banerjea, Shashwati, Pandey, Mayank, and Tyagi, Neeraj. Wireless Fog-Mesh: A communication and computation infrastructure for IoT based smart environments. In *Mobile, Secure, and Programmable Networking*, pages 322–338, 2019. DOI: 10.1007/978-3-030-03101-5_27.

[2] Bhatnagar, Nirdosh. *Mathematical Principles of the Internet*. CRC Press, 2019. ISBN: 9781138505483.

[3] Bonomi, Flavio, Milito, Rodolfo, Zhu, Jiang, and Addepalli, Sateesh. Fog computing and its role in the Internet of Things. In *Proceedings of the first edition of the MCC workshop on Mobile cloud computing*, pages 13–16. ACM, 2012. DOI: 10.1145/2342509.2342513.

[4] Bormann, Carsten, Castellani, Angelo P, and Shelby, Zach. CoAP: An application protocol for billions of tiny internet nodes. *IEEE Internet Computing*, 16(2):62–67, 2012. DOI: 10.1109/MIC.2012.29.

[5] Cheshire, Stuart and Krochmal, Marc. DNS-based service discovery. RFC 6763, RFC Editor, 2013.

[6] Cirani, Simone, Davoli, Luca, Ferrari, Gianluigi, Léone, Rémy, Medagliani, Paolo, Picone, Marco, and Veltri, Luca. A scalable and self-configuring architecture for service discovery in the Internet of Things. *IEEE Internet of Things Journal*, 1(5):508–521, 2014. DOI: 10.1109/JIOT.2014.2358296.

[7] Damgård, Ivan Bjerre. A design principle for hash functions. In *Conference on the Theory and Application of Cryptology*, pages 416–427. Springer, 1989. DOI: 10.1007/0-387-34805-0_39.

[8] Datta, Soumya Kanti and Bonnet, Christian. Search engine based resource discovery framework for Internet of Things. In *2015 IEEE 4th Global Conference on Consumer Electronics (GCCE)*, pages 83–85. IEEE, 2015. DOI: 10.1109/GCCE.2015.7398707.

[9] Datta, Soumya Kanti, Da Costa, Rui Pedro Ferreira, and Bonnet, Christian. Resource discovery in Internet of Things: Current trends and future standardization aspects. In *2nd World Forum on Internet of Things (WF-IoT)*, pages 542–547. IEEE, 2015. DOI: 10.1109/WF-IoT.2015.7389112.

[10] Eastlake, Don and Hansen, Tony. US secure hash algorithms (SHA and HMAC-SHA). RFC 4634, RFC Editor, 2006.

[11] Hunkeler, Urs, Truong, Hong Linh, and Stanford-Clark, Andy. MQTT-S—A publish/subscribe protocol for Wireless Sensor Networks. In *2008 3rd International Conference on Communication Systems Software and Middleware and Workshops (COMSWARE'08)*, pages 791–798. IEEE, 2008. DOI: 10.1109/COMSWA.2008.4554519.

[12] Jara, Antonio J, Lopez, Pablo, Fernandez, David, Castillo, Jose F, Zamora, Miguel A, and Skarmeta, Antonio F. Mobile digcovery: A global service discovery for the Internet of Things. In *2013 27th International Conference on Advanced Information Networking and Applications Workshops*, pages 1325–1330. IEEE, 2013. DOI: 10.1109/WAINA.2013.261.

[13] Jia, Bing, Li, Wuyungerile, and Zhou, Tao. A centralized service discovery algorithm via multi-stage semantic service matching in Internet of Things. In *2017 IEEE International Conference on Computational Science and Engineering (CSE) and IEEE International Conference on Embedded and Ubiquitous Computing (EUC)*, volume 1, pages 422–427. IEEE, 2017. DOI: 10.1109/CSE-EUC.2017.82.

[14] Jimenez, Raul, Osmani, Flutra, and Knutsson, Björn. Sub-second lookups on a large-scale Kademlia-based overlay. In *2011 IEEE International Conference on Peer-to-Peer Computing*, pages 82–91. IEEE, 2011. DOI: 10.1109/P2P.2011.6038665.

[15] Kamel, Mohammed B. M., Crispo, Bruno, and Ligeti, Peter. A decentralized and scalable model for resource discovery in IoT network. In *2019 International Conference on Wireless and Mobile Computing, Networking and Communications (WiMob)*, pages 1–4. IEEE, 2019. DOI: 10.1109/WiMOB.2019.8923352.

[16] Kamel, Mohammed B. M., Ligeti, Peter, Nagy, Adam, and Reich, Christoph. Distributed Address Table (DAT): A decentralized model for end-to-end communication in IoT. To appear in Journal of P2P Networking and Applications, 2021.

[17] Kamel, Mohammed B. M., Ligeti, Peter, and Reich, Christoph. Region-based distributed hash table for fog computing infrastructure. In *13th Joint Conference on Mathematics and Informatics*, pages 82–83, 2020.

[18] Lua, Eng Keong, Crowcroft, Jon, Pias, Marcelo, Sharma, Ravi, and Lim, Steven. A survey and comparison of peer-to-peer overlay network schemes. *IEEE Communications Surveys & Tutorials*, 7(2):72–93, 2005. DOI: 10.1109/COMST.2005.1610546.

[19] Maurer, Ward Douglas and Lewis, Theodore Gyle. Hash table methods. *ACM Computing Surveys (CSUR)*, 7(1):5–19, 1975. DOI: 10.1145/356643.356645.

[20] Maymounkov, Petar and Mazieres, David. Kademlia: A peer-to-peer information system based on the xor metric. In *International Workshop on Peer-to-Peer Systems*, pages 53–65. Springer, 2002. DOI: 10.1007/3-540-45748-8_5.

[21] Mokadem, Riad, Hameurlain, Abdelkader, and Tjoa, A Min. Resource discovery service while minimizing maintenance overhead in hierarchical DHT systems. *International Journal of Adaptive, Resilient and Autonomic Systems (IJARAS)*, 3(2):1–17, 2012. DOI: 10.4018/jaras.2012040101.

[22] Montresor, Alberto and Jelasity, Márk. PeerSim: A scalable P2P simulator. In *Proc. of the 9th Int. Conference on Peer-to-Peer (P2P'09)*, pages 99–100, Seattle, WA, September 2009. DOI: 10.1109/P2P.2009.5284506.

[23] Pahl, M. and Liebald, S. A modular distributed IoT Service Discovery. In *2019 IFIP/IEEE Symposium on Integrated Network and Service Management (IM)*, pages 448–454, 2019.

[24] Pahl, Marc-Oliver. *Distributed smart space orchestration*. PhD thesis, Technische Universität München, 2014. https://mediatum.ub.tum.de/1196145.

[25] Pattar, Santosh, Buyya, Rajkumar, Venugopal, KR, Iyengar, SS, and Patnaik, LM. Searching for the IoT resources: Fundamentals, requirements, comprehensive review, and future directions. *IEEE Communications Surveys & Tutorials*, 20(3):2101–2132, 2018. DOI: 10.1109/COMST.2018.2825231.

[26] Picone, Marco, Amoretti, Michele, and Zanichelli, Francesco. Geokad: A P2P distributed localization protocol. In *2010 8th IEEE International Conference on Pervasive Computing and Communications Workshops (PERCOM Workshops)*, pages 800–803. IEEE, 2010. DOI: 10.1109/PERCOMW.2010.5470545.

[27] Roos, Stefanie, Salah, Hani, and Strufe, Thorsten. On the routing of Kademlia-type systems. In *Advances in Computer Communications and Networks*. River Publishers, 2017.

[28] Rowstron, Antony and Druschel, Peter. Pastry: Scalable, decentralized object location, and routing for large-scale peer-to-peer systems. In *IFIP/ACM International Conference on Distributed Systems Platforms and Open Distributed Processing*, pages 329–350. Springer, 2001. DOI: 10.1007/3-540-45518-3_18.

[29] Stoica, Ion, Morris, Robert, Karger, David, Kaashoek, M Frans, and Balakrishnan, Hari. Chord: A scalable peer-to-peer lookup service for internet applications. *ACM SIGCOMM Computer Communication Review*, 31(4):149–160, 2001. DOI: 10.1145/964723.383071.

[30] Wirtz, H., Heer, T., Hummen, R., and Wehrle, K. Mesh-DHT: A locality-based distributed look-up structure for Wireless Mesh Networks. In *2012 IEEE International Conference on Communications (ICC)*, pages 653–658, June 2012. DOI: 10.1109/ICC.2012.6364336.

[31] Wirtz, Hanno, Heer, Tobias, Serror, Martin, and Wehrle, Klaus. DHT-based localized service discovery in wireless mesh networks. In *2012 IEEE 9th International Conference on Mobile Ad-Hoc and Sensor Systems (MASS 2012)*, pages 19–28. IEEE, 2012. DOI: 10.1109/MASS.2012.6502498.

[32] Yang, Yijun, Zhang, Xiaomei, Yu, Jianping, Zhang, Peng, et al. Research on the hash function structures and its application. *Wireless Personal Communications*, 94(4):2969–2985, 2017. DOI: 10.1007/s11277-016-3760-4.

[33] Zhao, Ben Y, Huang, Ling, Stribling, Jeremy, Rhea, Sean C, Joseph, Anthony D, and Kubiatowicz, John D. Tapestry: A resilient global-scale overlay for service deployment. *IEEE Journal on selected areas in communications*, 22(1):41–53, 2004. DOI: 10.1109/JSAC.2003.818784.