Peer-to-Peer Networking and Applications (2022) 15:178–193
https://doi.org/10.1007/s12083-021-01221-3

Distributed Address Table (DAT): A Decentralized Model for End-to-End Communication in IoT

Mohammed B. M. Kamel[1,2,3] · Peter Ligeti[1] · Adam Nagy[1] · Christoph Reich[2]

Received: 14 September 2020 / Accepted: 8 July 2021 / Published online: 24 September 2021

Abstract

To achieve a fully connected network in Internet of Things (IoT) there are number of challenges that have to be overcome. Among those, a big challenge is how to keep all of the devices accessible everywhere and every time. In the IoT network, the assumption is that each IoT device can be reached by any client at any given time. In practice, this is not always possible and without a proper mechanism the nodes behind a NAT are unable to communicate with each other directly, and their addresses have to be shared through a trusted third party. This challenge becomes harder by taking into consideration that most NAT traversal approaches have been developed prior to rising of the IoT, without taking into account the constrained nature of the participating devices and mostly depend on a centralized entity. In this paper we proposed the Distributed Address Table (DAT), a decentralized, secure and lightweight address distribution model that allows any two nodes to get the addresses of the other end without relying on a trusted third party. Structured Peer-to-Peer (P2P) overlay by utilizing Distributed Hash Table (DHT) technique is generated as its underlying communication scheme to ensure that all participating devices are accessible at any given time. This is achieved through simple, yet secure and efficient decentralized model. The DAT adopts the edge/fog computing paradigms to ensure a decentralized address distribution. The results showed that the proposed model is efficient. In addition, the security properties of the proposed model have been defined and proved.

Keywords Address Distribution · P2P · Decentralized Architecture · NAT Traversal · Self Adaptive Algorithm

1 Introduction

Although in the Internet of Things (IoT) network, the communication can be based on a URI (e.g. using CoAP protocol), but the exact communication is established and maintained by targeting the socket address (i.e. IP address and port number) of a specific resource as a result of lookup in the network. In a network where the used addresses of the communication are based on the Internet Protocol (IPv4) and due to lack of enough available addresses, they have been categorized in two main parts: global addresses that are accessible by all nodes and private addresses that are local to a small part of the network and interconnected with the rest of the network by a Network Address Translator (NAT) [10]. The nodes with global addresses can be easily accessed using their unique routable addresses. The nodes that reside behind a NAT can communicate with any node that has a global unique address, but the Internet architecture makes it difficult for nodes behind different NATs to communicate with each other directly. The reason is that the NAT allows only the outgoing traffic to pass through and open a session for each of the outgoing packet. Upon arrival, an incoming packet can pass through the NAT if there is an open session assigned to it. Therefore, the packets from a node outside the NAT or firewall and without prior opened session will be dropped and the communication will fail to be established. This issue is crucial and might cause partial lose of connectivity in networks which affects the success implementations

✉ Mohammed B. M. Kamel
mkamel@hs-furtwangen.de

Peter Ligeti
ligetipeter@inf.elte.hu

Adam Nagy
nagy.adam@inf.elte.hu

Christoph Reich
christoph.reich@hs-furtwangen.de

[1] Eotvos Lorand University, Budapest, Hungary

[2] Hochschule Furtwangen University, Furtwangen im Schwarzwald, Germany

[3] University of Kufa, Kufa, Iraq

of models such as resource discovery [17, 18]. While using IPv6 solves the problem of lack of addresses, but still due to usage of IPv4 by a huge number of devices a NAT is required [32] to translates these two versions of addresses.

There are number of NAT traversal approaches that proposed number of solutions to make the NATed devices accessible. Some of these approaches [8, 16, 22, 29, 30] require a centralized entity to organize part of the process. Although relying on a rendezvous third party server helps organizing the distribution of addresses, but the centralized entity may turn into the bottleneck in the network. In addition, relying on a NAT traversal approach that requires a continuous keep-alive message and due to its required power is not feasible to be implemented in a network with constrained devices. Using protocols such as Port Control Protocol (PCP) [34] allows the devices in the network to add a port forwarding rule and therefore reduce the required power to be more suitable in a network with constrained devices. However, in PCP there is still a challenge on how to inform the other party about the details of the created information (i.e. IP address, protocol, and port), as the PCP itself does not provide this functionality. The issue of address distribution for end-to-end communication can be addressed in a centralized third-party entity. The PCP suggests centralized approaches such as a rendezvous server or DNS Update in DNS based service discovery [6]. Recently, researchers proposed models in which the centralized entity is replaced by a set of proxies combined with the cloud server [19] or by the distributed ledger technology [21].

According to Mamdouh et al. [23] one of the IoT vulnerabilities that can be used to attack the network is through the NAT traversal approaches. It can be used to permit remote attackers to achieve connection into network purposes like interconnection compromise other equipment, penetrate data or inject interfered or fake information to the network. In the process of removing the trusted third-party for address distribution and distributing the task among many nodes, the communication address has to be kept confidential and accessible only by the trusted entities in the IoT network. Therefore, in this paper we address the following research question: *How can the communication addresses be distributed with other nodes, considering the constrained nature of IoT devices and without involving a trusted third party or revealing the distributed addresses to the public?*

In this paper, we proposed the Distributed Address Table (DAT) that distributes the communication addresses of different nodes in the network in a decentralized scheme, taking into consideration the limited computing power of the IoT devices. Our proposed model adopts the edge/fog computing paradigm [33] and aims to remove centralized third parties by involving the edge and fog nodes in the process of communication. The model uses a structured peer-to-peer (P2P) scheme by applying Distributed Hash Table (DHT) [36] technology as the underlying scheme of communication. The main contributions of our paper are: (1) DAT creates a decentralized overlay for address distribution and does not depend on a trusted third-party entity. Therefore, it removes any possible single point of failure and attack during address distribution in the system. (2) Authorized nodes in the network and using DAT are able to distribute and get the addresses of nodes behind the NAT without knowing their current IP addresses. (3) The addresses in DAT are distributed securely and only the authorized nodes have access to a specific address in DAT, therefore the privacy of the distributed addresses are preserved. The rest of paper is organized as follows. The preliminaries will be introduced in the second section for a further understanding. In the third section, we will discuss the related work of address distributions. The fourth section explains our modelling and solution. The fifth section includes the evaluation and discussion of our proposal. Finally, the sixth section summarizes our model and the achieved results.

2 Preliminaries

2.1 NAT Traversal Protocols

The three main types of NAT traversal approaches are hole punching, relaying, and port forwarding. Session Traversal Utilities for NAT (STUN) [35] was among the first studies introduced as an approach for the NAT traversal, which was based on establishing a connection between two nodes with private addresses behind NAT using what so called *hole punching*. Hole punching assumes that the two communicating nodes already have active UDP sessions with a rendezvous server. The known and common server registers the communication addresses (i.e. IP and port) of each of the peers. Later, in the hole punching the addresses will be passed to the other party of the communication through the rendezvous server. This approach had limited applicability with some systems and as a result a probability of failure.

Traversal Using Relays around NAT (TURN) [27] was an alternative technique that tried to reduce the failures that occur in the STUN. In this approach, the NATed nodes create two client/server communications through *relaying*. In this case the two communicating nodes firstly establish the communication with a previously agreed server that has a unique public IP. Since both nodes behind the NAT started a communication to the server and a session already has been established in the corresponding NAT, the incoming packets from the server to each of them will be forwarded. In this case the transmitted packet goes to the common server first. Then, the packet will be forwarded to the receiver through already established communication channel. Although relaying solves the failure ratio issue in hole punching, but since the whole traffic pass through a common server it is least effective used technique due to its slow procedure. Interactive connectivity establishment (ICE) [28] introduces as a protocol that uses both hole punching and relaying techniques. It is based on STUN and TURN protocols and chooses the best possible solution among them. Keep-alive message is sent by the nodes to keep the port with the rendezvous server open and assigned.

In *port forwarding* approach, no rendezvous server is used by the communicating nodes. The nodes instruct the NAT devices to open a port and forward the open port to their local ports. In this case the incoming packets to the public ports will be forwarded to the local port of the private NATed nodes. The implemented protocols such as Internet Gateway Device protocol (IGD) that is based on Universal plug and play (UPnP) [4] specification propose an automated approach to allow the incoming traffic to be forwarded to a specific port. NAT Port Mapping Protocol (NAT-PMP) [7] has been implemented as an alternative to IGD, and is also based on UPnP. Its main drawback is supporting of one external IP address. PCP [34] has been designed as a successor to NAT-PMP and improved the protocol, which supports multiple external IP addresses. While no rendezvous server is required in port forwarding, but a third-party server is used to share and inform the other communicating node about the IP and open ports.

2.2 NAT Traversal in IoT Light Weight Protocols

There are two light weighted and most popular standard protocols for constrained IoT devices, namely Message Queuing Telemetry Transport (MQTT) [14] and Constrained Application Protocol (CoAP) [3]. MQTT is based on the publish/subscribe approach and facilitates one to many communications through a common node (i.e. broker). Clients publish the messages by sending them to the broker and subscribe for a specific message in the broker. The topics are categorized by different labels to distinguish them from each other. The Quality of Service (QoS) in MQTT can be described in three transport Quality of Service types: in QoS-0 the client sends the message to the broker and does not expect any confirmation of the receipt by the broker. Same with the receiving client, the broker send the message to the subscribed clients without any need to the acknowledgments. This called the Fire-and-Forget. In QoS-1 the broker sends a confirmation to the published client after receipt of the data. Similarly the subscribed client sends the acknowledgment upon receiving the message. In QoS-1 the confirmations send at least once, and if the expected acknowledgment has not received after the time-out it will be resent. In QoS-2, the messages send exactly once to prevent the duplicate in the routing.

CoAP [3] on the other hand, is a document transfer protocol (similar to HTTP) that works on a client-server architecture. Although it resembles the HTTP, but it has been designed taking into consideration the limited resources of the constrained devices with smaller packets comparing to HTTP. While in MQTT the broker could cause a single point of failure, in CoAP the participating applications form the clients and servers in the protocol. The support of CoAP to multicast and the lower power consumption makes it a better choice in the network with huge number of constrained devices.

CoAP devices can act as a client or server. CoAP servers has to be reachable by other devices in the network. If the CoAP server has its public IP then, communication can be established directly. Otherwise, if it is behind a NAT or firewall, the incoming communications traffic will be filtered by the NAT server which cause the failure of connection to the CoAP server. One solution can be use of a common server to establish a communication by punching a hole in the NAT which needs a centralized known server. Lightweight M2M (LWM2M) [31] is a lightweight device management that is used with the CoAP. Still, one of the main challenges is how to reach the devices behind NAT. Another solution is to keep the communication alive by sending a pulse [5] periodically. It is not feasible to the resource constrained IoT devices to send these periodic pulses. Similar to the hole punching it consumes unnecessary resources and produces latency. The reason is that in order to keep the NAT binding the endpoint device has to send periodic pulses, which consumes the constrained network resource as well as the battery of this constrained IoT device.

PCP [34] allows the IoT devices to control a NAT by creating a binding of their private IP and port. When PCP protocol is used, there is no need for an IoT device to send pulses periodically in order to keep the binding alive which makes it more suitable to use by the constrained devices. In addition, as illustrated in Fig. 1 even if an IoT device is located behind multiple NATs still it can create the binding through these NATs. The concern of this protocol is how to inform other peers about the new created communication data. The challenge of sharing the addresses between NATed nodes is that by targeting a destination IP no direct traffic is allowed to pass through the NAT and access the NATed node. In PCP the addresses are informed to the remote hosts using a centralized entity [34]. The problem of adopting these approaches in the IoT network can be summarized as follows:

- *Bottleneck & Single Point of Failure through central server*: The NAT traversal approaches need a common and previously determined server in order to be able to establish the connection behind NAT. While in PCP the actual communication does not require a centralised

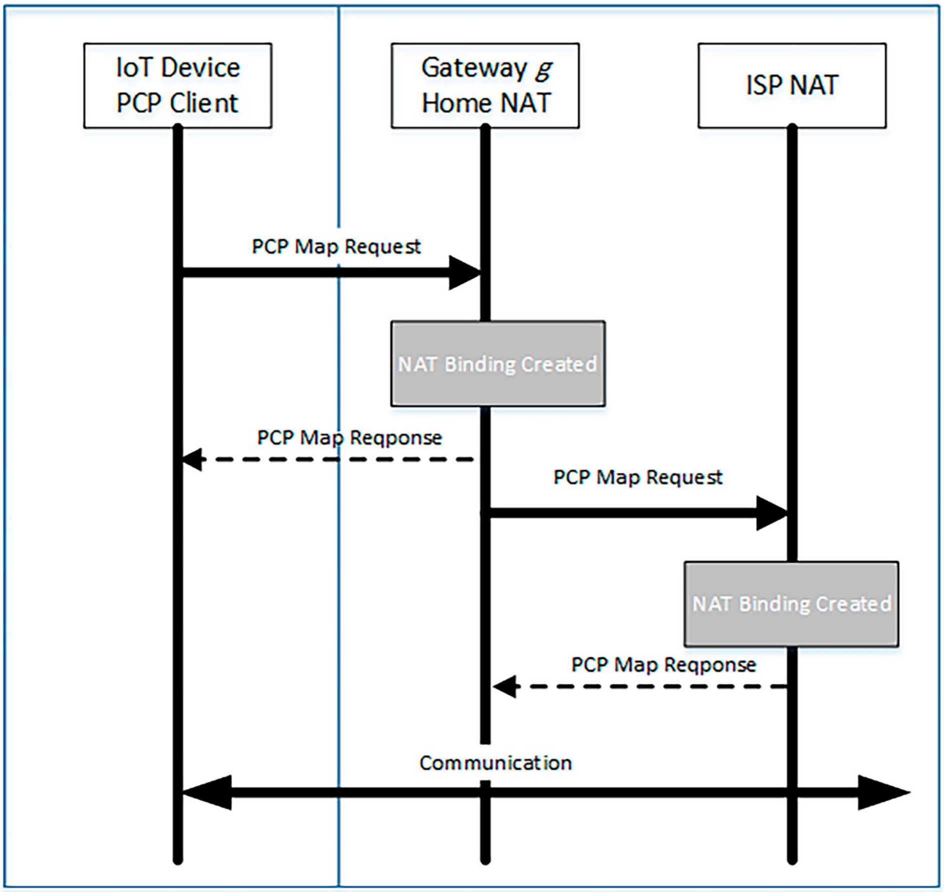

Fig. 1 Port Control Protocol (PCP)

server, but the opened ports and IP addresses are shared using a centralised entity. The third party entity might turn into a bottleneck of the system that affects its availability.

- *High Computation Power per Node*: Most NAT traversal approaches require high computation and energy power of the participating devices. As instance, in hole punching approach in order to keep the port alive there should be a continuous pinging each few seconds. The nature of the constrained devices makes this method inefficient in a network with devices that have limited computation power.

3 Literature Review

The distribution of addresses is among main challenges in different fields such as NAT Traversal approaches. The NAT Traversal approaches are not required only in the IoT network. There are different proposed approaches that target the problems related to the devices behind a NAT, such as the applications over LTE networks [1], social networks [8] and virtual desktops in distributed environments [11]. The NAT traversal and their address distribution models can be divided into three main approaches: The centralised approach that uses a rendezvous server, the hybrid approach that in addition to a rendezvous server uses other distributed nodes, and decentralised approach that removes single third party entity from the system.

There are number of researchers that proposed the NAT traversal approaches using a rendezvous server known to both communicating nodes. Authors in [29] presented an approach employed UDP sessions with a rendezvous protocol server to communicate peer information and TCP sessions established in NAT hole punching. Cho [8] developed synchronization of contact information used in social networks through UDP hole punching as well as implementation of unstructured P2P in Wireless Area Network (WAN) platform. Their proposed model uses a rendezvous server known to both parties running an Android operating system. Stephenson and Namiot [30] demonstrated a comparative analysis of data communication of mobile web clients and showed how the real-time Communication (WebRTC) [16] utilizing P2P scheme needs a known server in order to establish the sessions between NATed clients. Lyu et al. [22] developed a Netlet middleware to simplify the programming of applications that are based on P2P architecture. It contains a hole punching approach to allow access to the NATed application, using TCP and UDP servers. In these models the address of public rendezvous server is known by both sender and the receiver and the third party server is responsible for the distribution of addresses and the establishment of communication between the two communicating parties.

Kavalionak et al. [19] proposed an approach called NAT-Cloud to solve the issue of distributing the communication addresses and accessing the NATed peers in the P2P applications. Their proposed hybrid approach in addition to the cloud computing uses publicly available peers in the P2P network for address distribution. A node in their proposed approach chooses a list of proxy nodes from public proxy servers or a rented cloud. The communication addresses of the nodes in the proposed model is distributed using one of the proxy nodes in their list. Although each node has to send a keep-alive message periodically to the public proxy servers in its list, but this is not a requirement with the rented cloud servers. Herry et al. [13] proposed a secure update framework with a P2P overlay that removes the centralized update server. For NAT traversal, the proposed framework uses a hole-punching approach that enables the NAT-bindings via a specific port to allow the client messages communication using a common known server in the framework. In [20] the authors developed a gossip peer sampling protocol called *Nylon* that is able to access the NATed node in the network. A NATed node communicates with any neighbor node through a reactive hole-punching by creating a path of nodes that work as relays. The node can rely on one or more relays in order to be able to access the destination peer.

Kfoury et al. [21] proposed a decentralized and blockchain based approach to distribute the addresses that have been created in NAT binding. This method uses smart contracts offered by Ethereum[1] to distribute the addresses of NATed nodes. The strengths of this approach are removing the trusted third party needed by similar approaches and distributing the trust among many nodes using blockchain technology. The use of Ethereum removes the need to a centralized rendezvous server, but it the distribution of addresses is achieved with some cost, called Ethereum Gas[2]. The need to a centralized rendezvous server, continuous keep-alive messages or payable cost to distribute the addresses might affect the adoption of the approach considering the distributed nature of the IoT network, the resource constrained IoT devices and frequent updates of the communication addresses. Our primary focus in this paper is to develop a distributed address table that works in a decentralized manner to inform the peers about the communication address of an IoT device without any need to a centralized authority taking into consideration the limited available resources of devices in the IoT network and the privacy of the participants.

4 Model Description

An important characteristic of a protocol designed for IoT is the avoidance of single point of failure as it can be a centralized service, even if implemented using redundancy and replication. This can be achieved through the adoption of edge/fog computing. The nodes in the edge/fog computing layer can be any resource-rich device such as gateways and network routers. Due to the distributed nature of fog layer (comparing to the centralized nature of cloud layer) the challenge is how to organize the fog nodes in fog computing. In our proposed model and using the DHT technology we created a distributed address table (DAT) overlay of peers without any centralised entity. DAT is based on the Kademlia implementation [24] of DHT. It stores a quadruple of data in the overlay and has a unique method of identifier generations. The reason behind using P2P scheme in this model is its distinguished features that fit the IoT requirements. P2P scheme can be easily scaled without any need to a reorganizing and synchronizing authority. Since the information are distributed among many edge and fog nodes, there is no single point of failure and breach or shutting down a small subset of peers does not affect the overall availability.

As mentioned earlier, the solution that introduces for the IoT environment has to take into consideration the limited computation power of the participating devices and the required distributed scheme of the proposed approach. As a result, the approaches that require a centralized known entity and heavy computation power including relaying the traffic through or keep a connection alive by continuous pinging are hard to adopt in the IoT network due to the discussed limitations of the IoT devices.

Figure 2 shows the structure of the distributed address table. On the left side there is a set of nodes $\mathcal{N}$ that are physically located in different parts in the IoT network. A node $n \in \mathcal{N}$ can be an IoT resource or a client in the IoT network that reside behind a NAT or firewall. Due to the mechanism of the NAT, two nodes in $\mathcal{N}$ behind different NATs can not create a direct communication without a prior binded port in the corresponding NAT. Otherwise, the incoming direct traffic will be dropped. Each subset of nodes in $\mathcal{N}$ are connected to an IoT edge/fog node (e.g. an IoT gateway) $w \in \mathcal{W}$ that is located on the right side of the structure. A gateway's responsibility may vary from handling a few nodes (e.g. smart home) to hundreds of nodes (e.g. public fog node or environmental monitoring). The sets $\mathcal{N}$ and $\mathcal{W}$ are disjoint sets. At any given time the

[1] https://ethereum.org/en/developers/docs/smart-contracts/

[2] https://ethereum.org/en/developers/docs/gas

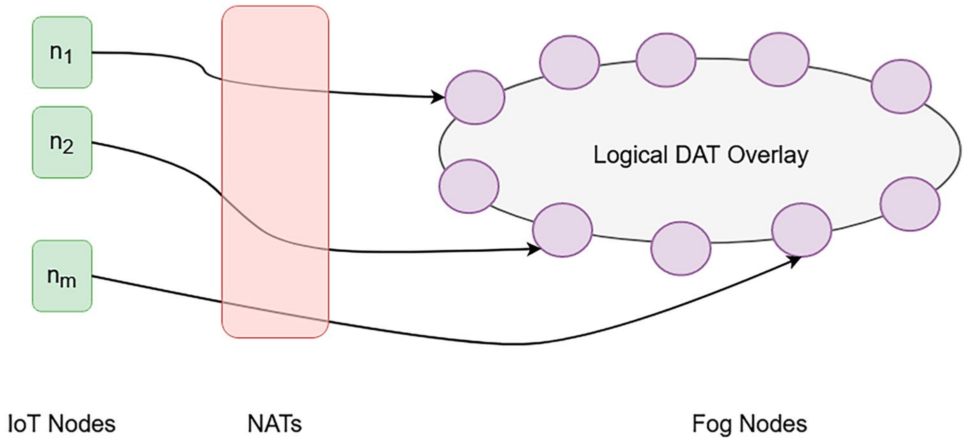

Fig. 2 Distributed Address Table Structure

relationship between the members of $\mathcal{N}$ is many-to-one with members of $\mathcal{W}$. i.e. one or more $n \in \mathcal{N}$ is connected to the IoT network through a $w \in \mathcal{W}$. The proposed model uses a structured P2P between gateways to create the DAT overlay that provides a structured method of distribution and lookup of addresses in the network. In the following sections, the participants of the model, their relationship and responsibilities are explained in detail.

4.1 Model Components and Assumptions

Parameters The first segment of participants of the model are represented as a finite set of nodes $\mathcal{N} = \{1,2,\ldots,j\}$ in the IoT network. Every node is assigned to a gateway $w \in \mathcal{W}$. Let $H(.)$ be a hash function, $Enc_k(m)$ be an encryption of the message m using symmetric key k and $Sign_w(p)$ be a digital signature for packet p generated by $w \in \mathcal{W}$ gateway.

Participants There are four different roles and so type of participants in the model where one wants to share its address: issuer, receiver (friend), gateway and regular nodes. The roles of these types of participants are according to the following explanation. It is possible that the communication address of each node changes, for e.g. as a result of joining a different physical network. The node that its address has been changed recently is known as *issuer node*, represented by $i \in \mathcal{N}$ further on. It will start informing its *friend nodes* ($\mathcal{F}_i \subset \mathcal{N} \setminus \{i\}$) directly of its new address. A node $f \in \mathcal{F}_i$ might not be able to receive the address of i, e.g. f is currently offline or the incoming traffic from i has been dropped by the connected NAT of f. Any friend node that did not acknowledge the data transmitted by the issuer node will be added to the uninformed node set. The issuer node generates a quadruple for each of the uninformed nodes that includes information on the issuer, the receiver and some encrypted data. The

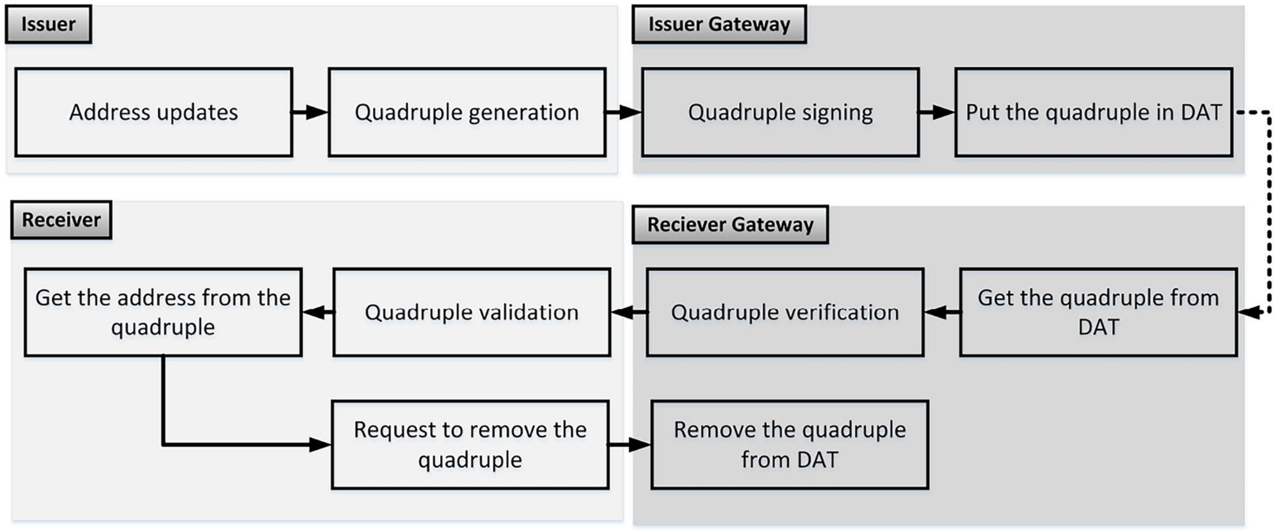

Fig. 3 DAT Model Workflow

generated quadruple will be signed by its directly connected *gateway* and added to the DAT. Later, a *friend node* will be informed by getting the corresponding quadruple through its connected *gateway*. After receiving the quadruple and access the *issuer* node, the quadruple can be removed from DAT. Figure 3 illustrates the workflow of DAT model.

The nodes that are not issuer, receiver or engaged gateway nodes, are known as *regular nodes*. Furthermore, the participants have private identifiers, which are supposed to be permanent values and are known only by the friends and the participant itself. We can assume the participants differ in the computational power from each other as well.

Communication model As we noted above, a structured P2P network is supposed to exist between gateways, i.e. the peers in the DAT overlay are members of $\mathcal{W}$. This network is described by a graph $G_{\mathcal{W}} = (\mathcal{W}, E)$. Behind each peer w there are one or more connected devices (belong to $\mathcal{N}$). Additionally, a connection is supposed between a participant and its friends, i.e. every participant i has a set of its friends $\mathcal{F}_i$ to communicate with in a secure and trusted way. Note that the members of a friend set $\mathcal{F}_i$ of a node i are connected to each other through members of $\mathcal{W}$ but they are not part of the DAT overlay itself.

Security model Before defining the security properties, we need to define some natural properties which are necessary for the model to function. Regarding nodes we need to assume that from the viewpoint of any node $i \in \mathcal{N}$, the set of friends $\mathcal{F}_i$ are *honest* nodes. The rest of the nodes $\mathcal{R}_i = \{r \in \mathcal{N} \setminus \mathcal{F}_i\}$ can be assumed to be *malicious*. In the case of gateways, we have to assume that there is no cut in $G_{\mathcal{W}}$ containing malicious gateways only (otherwise, the malicious gateways together could make the communication impossible between nodes). More precisely, we assume that for a given node i for every friend $f \in \mathcal{F}_i$ there exist a path $(w_1, e_1, w_2, \dots, w_k)$ such that the issuer i is connected to w_1, the receiver f is connected to w_k and all of $w_1, \dots, w_k$ are *semi-honest*. The semi-honest entities are assumed to follow the protocol properly, but they are allowed to store the received data locally in an attempt to get more information from the stored data. The remaining gateways are supposed to be *malicious*.

Beside these properties, the nature of the communication model also regulates the applicable security. In this model only the gateways are able to use public key cryptography, while the nodes and because of their limited computation power can encrypt and decrypt messages using symmetric keys only. In case of 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 computational security, i.e. we assume probabilistic polynomial-time (PPT) adversaries with negligible success probabilities. A function has a negligible success probability if it occurs with a probability smaller than any polynomial fraction [2]. The goal of the security model is to allow friends to securely and anonymously distribute their addresses to be able to communicate with each other, which in our model comes from the following security properties:

- **Weak anonymity**: A PPT adversary can learn any connection between a given quadruple and the friend identifier of the sender or the receiver with negligible probability only.
- **Address privacy**: PPT gateways and non-friend nodes can learn the distributed address from a given quadruple with negligible probability only.
- **Soundness**: A PPT adversary is able to generate a valid quadruple on behalf of an honest user with negligible probability only.

4.2 Setup phase

Each edge/fog node $w \in \mathcal{W}$ has a public identifier id_w. We suppose that these identifiers are chosen uniformly at random from a given range, e.g. from bit strings of length 512. The identifier of $w \in \mathcal{W}$ indicates its virtual location in the DAT. Depending on its virtual location, it will be responsible to store part of the generated quadruples by members of $\mathcal{N}$. Each node $i \in \mathcal{N}$ has a private identifier ID_i. This identifier is known only by the members of the friend set $f \in \mathcal{F}_i$ of node i. In addition, for every node i and for each $f \in \mathcal{F}_i$ a common secret key (k_{if}) is generated and shared between them on a secure channel. The key k_{if} is used to encrypt the transmitted data between these two nodes. 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. Every gateway $w \in \mathcal{W}$ can generate a digital signature $Sign_w(p)$ of any transmitted packet p. After an update in the communication address (e.g. a new address has been assigned to the node), the node i starts to inform members of $\mathcal{F}_i$ directly of the newly updated communication data. If a currently online node $f \in \mathcal{F}_i$ is accessible, it directly receives the packet. Otherwise, if the node f is offline or is behind a NAT that drop the incoming packets, the node f will not receive the packet. In this case and for each unacknowledged node f, a specific quadruple will be issued in order to guarantee that the current communication address will be received by object f.

4.3 DAT Quadruples

The DAT overlay consists of peers that are responsible to store a number of quadruples. These quadruples and their structures are explained in this section. For each uninformed friend node $f \in \mathcal{F}_i$, the node i creates a DAT quadruple

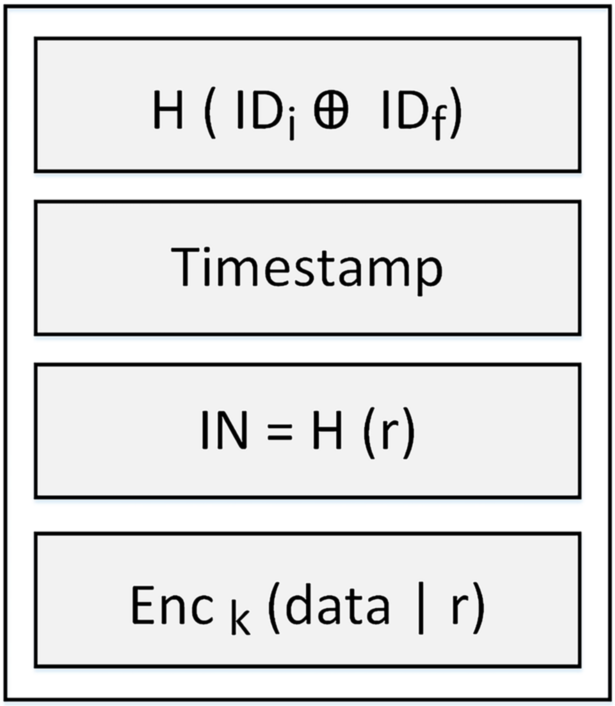

Fig. 4 DAT Quadruple Structure

consisting of four sections in this order: the common identifier (ID_{if}), timestamp, identification number (*IN*) and the encrypted data as shown in Fig. 4. The generation of DAT quadruple is explained in Algorithm 1.

```
Algorithm 1: Pseudo-code of generating the DAT quadruple
Input:
$id_i$: the private identifier of node $i$
$id_f$: the private identifier of node $f$
$data$: the communication data
$k_{if}$: the common key of node $i$ and $f$
Output: $DATQ$: the DAT quadruple
$header \leftarrow H(ID_i \oplus ID_f)$
$ts \leftarrow time$
$r \leftarrow RNG$
$IN \leftarrow H(r)$
$c \leftarrow Enc_{k_{if}}(data|r)$
$DATQ \leftarrow (header|ts|IN|c)$
return $DATQ$
```

The DAT quadruple will be constructed as following:

- *header*: Contains the common identifier of nodes i and f, ID_{if}. It will be used by *f* to indicate that this specific quadruple belongs to it. This section is computed by first xoring the IDs of *i* and *f*, then hashing the result.
- *timestamp*: In order to prevent impersonate of issuer node, a timestamp (TS) will be added. Adding the TS prevents specific types of attacks such as replay attack.
- *Identification Number*: The Identification Number (*IN*) will be used to proof the ownership of this specific quadruple at future communications. A (long enough) random string *r* is generated by the issuer of the packet during quadruple generation using the Random Number Generator (RNG). This string is resided as part of the data section that will be encrypted. The hashed value of the chosen random string *r* is called the Identification Number. It is used later on as part of the DRP (see Sect. 4.4) to proof the ownership of the packet by revealing the pre-image of *IN* (i.e. *r*) and therefore remove this quadruple from the DAT overlay in case of receiving it by the receiver node *f*.
- *Encrypted data*: The last part includes the encrypted data of the issuer node *i*. Along with the communication address, the chosen random string *r* is concatenated. The common secret key k_{if} is used to encrypt this section.

Each edge/fog node $w \in \mathcal{W}$ in DAT overlay is responsible to store a number of generated quadruples. Let $\mathcal{I}_d$ be a set of all possible sequences of d-bit binary digit (i.e. identifiers) and each $w \in \mathcal{W}$ has an identifier $id_w \in \mathcal{I}_d$ and a quadruple q has a $header_q \in \mathcal{I}_d$. Let define the following set of peers

$$M(q) = \{w : header_q \approx id_w, \nexists w' \mid dst(id_{w'}, header_q) < dst(id_w, header_q)\} \tag{1}$$

Where *distance function (dst)* is the distance between any two given identifiers. The model does not depend on a specific function to compute the closeness, and any particular distance function can be used. The set $M(q)$ links a quadruple q in DAT depending on its header $header_q$ to a peer $w \in \mathcal{W}$ in DAT that its identifier $id_w \in \mathcal{I}_d$ is close or equal to $header_q$. The cardinality of $M(q)$ depends on the *replication factor (rp)*. The replication factor indicates the number of close peers to w that are also responsible for storing a replica of the quadruple in DAT. The *replication factor* is chosen such that all the members in any given subset of $\mathcal{W}$ with cardinality rp are unlikely to fail (i.e. being inaccessible or offline).

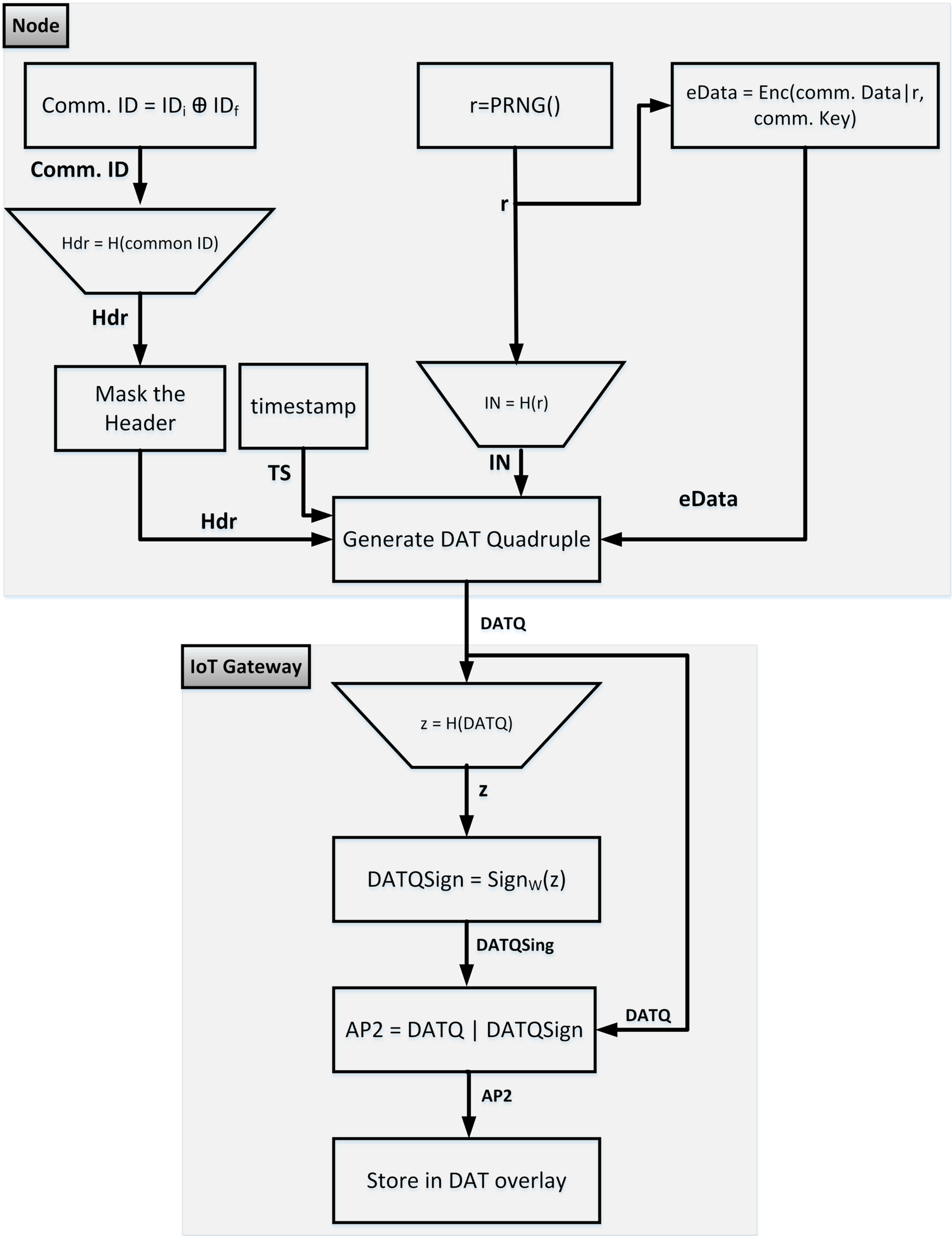

Fig. 5 Adding a DAT quadruple to the DAT overlay

4.4 The Control Packets

There are three types of control packets the participating nodes transmit in the network. Address Propagator Packet (AP2) to add a quadruple in the DAT, Address Request Packet (ARP) to get a quadruple from the DAT, and Dropping Request Packet (DRP) to remove a quadruple from the DAT overlay.

Address Propagator Packet (AP2) After generating the DAT quadruple by node i, it will be sent to the directly connected gateway w. The DAT quadruple will be signed by w to form the *Address Propagator Packet (AP2)* and sent to be stored in the DAT overlay (Fig. 5).

To keep the sender address private, a mask value α is used. The α parameter sets at the system setup and determines the number of bits that will be filtered from the header part of *AP2*. As instance, if α parameter is set to four, it means that last four bits out of the d-bit long header will be filtered and the packet will be stored in the peers that have same first $d-4$ bits prefix in their identifiers (i.e. 2^4 peers).

The node i applies the α parameter on the hashed value of the common identifier $ID_{if} = ID_i \oplus ID_f$ to filter the precise header address. Then, it adds the time stamp and the identification number, encrypts the communication address that resides in the data part using the common pre-shared key k_{if} and sends the quadruple to the connected gateway. After signing the packet, gateway w sends the *AP2* to edge/fog nodes in the DAT overlay that reside within the resulted subset of addresses. The procedure of calculating the recipient filtered address is illustrated in Algorithm 2.

Algorithm 2: Pseudo-code of calculating the recipient filtered address

```
Input:
    $\alpha$: the predefined address mask
    $d$: number of bits of the messages digest
    $ID_i$: the private identifier of node $i$
    $ID_f$: the private identifier of node $f$
Output:
    $fa$: the resulted filtered address
$ca \leftarrow H(ID_i \oplus ID_f)$
$prefixBits \leftarrow d - \alpha$
$fa \leftarrow MSB_{prefixBits}$ of $ca + \alpha$ times 0
return $fa$
```

Address Request Packet (ARP) Directly after becoming online in the network, a node f requests to receive any stored quadruple in the DAT with the specific header h as shown in Eq. 2. The node f can get the updated communication address of node i by requesting to get the corresponding DAT quadruple from the overlay (Fig. 6). This procedure guarantees a node knows about current communication addresses of the members of $\mathcal{F}$ securely without any centralized entity. In addition to weak anonymity (theorem 1), if the precise header is filtered (i.e. $\alpha > 0$), the privacy of nodes will be guaranteed.

$$h = \begin{cases} H(ID_i \oplus ID_f) & \alpha = 0 \\ MSB(d-\alpha, H(ID_i \oplus ID_f)) \sqcup \{0\}^{\alpha} & \alpha > 0 \end{cases} \qquad (2)$$

Dropping Request Packet (DRP) After receiving an *AP2* by a corresponding peer resides in the DAT overlay, it will store the quadruple locally for a specific period of time depending on the caching expiry parameters. If the receiver (i.e. the friend node) receives the quadruple and gets the updated communication address of the sender, there is no need for that specific quadruple to remain in the DAT overlay. A quadruple can be removed from the overlay prior to its expire time if a DRP is issued. Each friend $f \in \mathcal{F}_i$ after receiving its quadruple which includes the updated communication data of node i and verifying its content will issue a DRP packet in order to remove that quadruple from the DAT overlay. The DRP includes two parts: ID_{if} and the pre-image of *IN* parameter (i.e. r). After receiving a DRP by the IoT gateway in DAT overlay and verifying its validity, the particular quadruple will be removed from the local storage. The verification is done by checking the hashed value of the pre-image of *IN* parameter, which has to be equal to the *IN*.

5 Evaluation

5.1 Security Analysis

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

Proof Suppose that the quadruple

$$Q = (Q_1|Q_2|Q_3|Q_4) = (H(ID_i \oplus ID_f)|TS|H(r)|Enc_{k_{if}}(data|r))$$

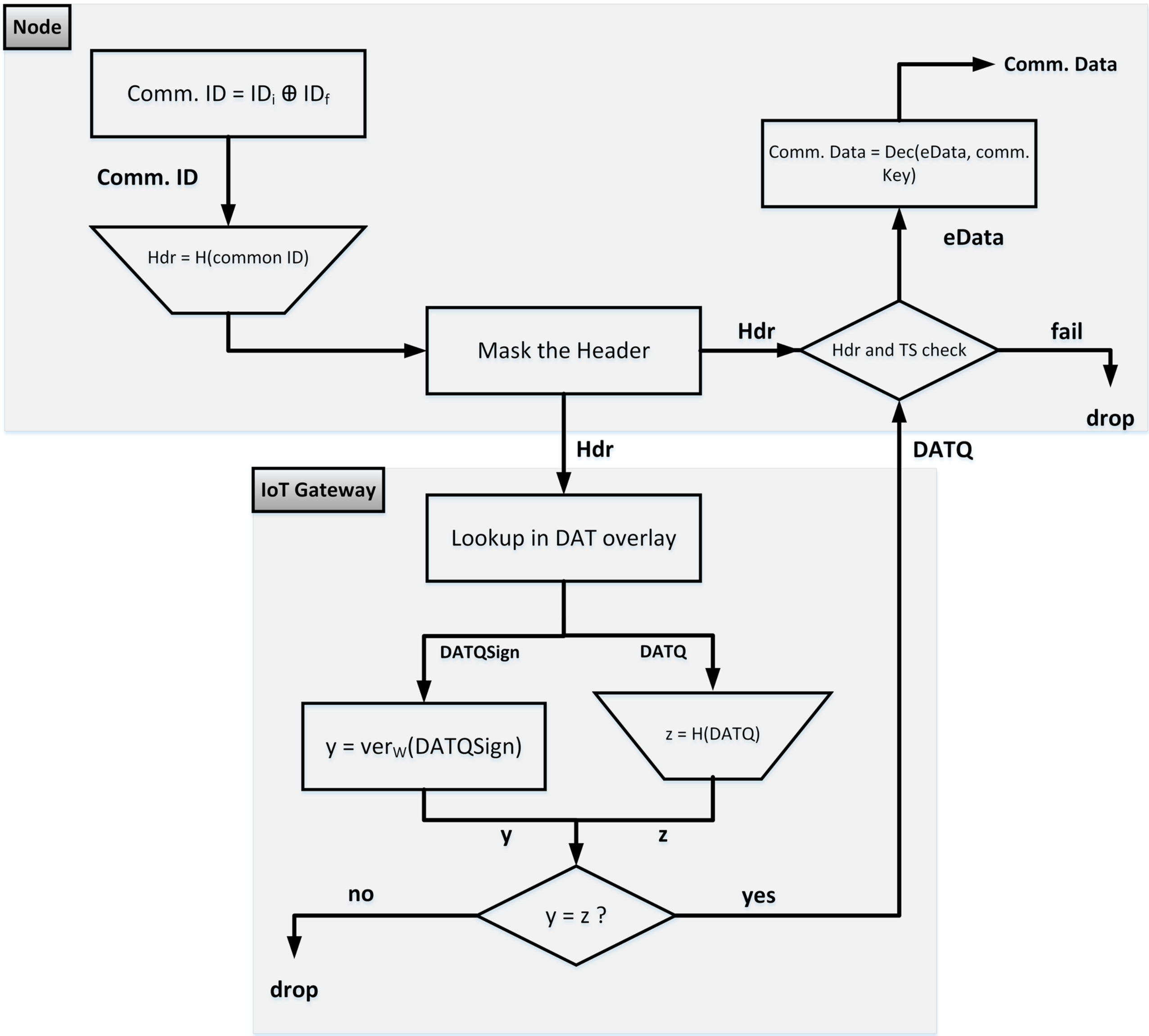

Fig. 6 Getting a DAT quadruple from the DAT overlay

is sent by node i to its friend $f \in \mathcal{F}_i$. Note that, only Q_1 includes on some information related to both the sender and the receiver (i.e. ID_i and ID_f), hence we can deal with this part of the packet only. First suppose that a node $m \in \mathcal{N} \setminus \mathcal{F}_i$ 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 preimage of $H(ID_i \oplus ID_f)$, then she can compute ID_i. However, since $H()$ is a one-way function, m can find any x with $H(x) = Q_1$ with negligible probability only. The gateways and the remaining nodes outside $\mathcal{F}_f$ are in a much hopeless situation, since even if they are assumed to find a preimage of the hash, after that they have to remove ID_f which is chosen randomly arising unconditional weak anonymity in this case. This completes the proof.

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

Proof Suppose that the quadruple

$$Q = (Q_1|Q_2|Q_3|Q_4) = (H(ID_i \oplus ID_f)|TS|H(r)|Enc_{k_{if}}(data|r))$$

is sent by node i to its friend $f \in \mathcal{F}_i$ and a malicious node/gateway $m \in \mathcal{W} \cup (\mathcal{N} \setminus \mathcal{F}_i)$ wants to learn the updated communication address (i.e. *data*). Note that, only Q_4 depends on the updated communication address, hence we can deal with this part of the quadruple only. The *data* is part of the plaintext that has been encrypted with a computationally secure encryption. Therefore, *data* and without the knowledge of

Table 1 Network parameters

type	parameter
local connection latency	2 ms
intra-regional latency	10 - 30 ms
long distance latency	80 - 120 ms

the symmetric key k_{if} 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 soundness.*

Proof Suppose that the quadruple

$$Q = (Q_1|Q_2|Q_3|Q_4) = (H(ID_i \oplus ID_f)|TS|H(r)|Enc_{k_{if}}(data|r))$$

is sent by node i to its friend $f \in \mathcal{F}_i$ and the adversary controlling a malicious node/gateway $m \in \mathcal{W} \cup (\mathcal{N} \setminus \mathcal{F}_i)$ wants to update the original communication address. Suppose that a malicious gateway can collect the set of quadruples $\mathcal{Q}$ with the same header (Q_1) and share it with m. To update the original communication address, the malicious node m has to replace the part containing information related to *data*, i.e. Q_4 and maybe the identification number Q_3, because Q_4 includes r as well. First, suppose that m wants to forge Q_4 only (i.e. the *data*). Since *Enc* is a computationally secure encryption, then *data* and r can be computed from a Q_4 part with negligible probability only. In addition, it is possible to compute new invalid communication address $data'$ concatenated with r in the original Q_4 (i.e. $Q'_4 = Enc_{k_{if}}(data'|r)$) with negligible probability only. Since $H()$ is a one-way function, it is hard to find r from Q_3. Now suppose that m is able to compute invalid quadruple parts Q''_3, Q''_4 with $Q''_3 = H(r''), Q''_4 = Enc_{k_{if}}(data''|r'')$. Since r'' can be computed with negligible probability from the cipher-text part, m has to find any string x'' with the same image $H(r'')$ which can be done with negligible probability as a consequence of collision resistance. This completes the proof. □

Corollary 1 *If H(.) is a collision-resistant one-way hash function and Enc is a computationally secure encryption, then the system satisfies every security properties together.*

5.2 Experimental Evaluation

In order to study the performance of DAT and validate its feasibility and reliability, several issues such as required preparation time for address distribution in constrained IoT devices and the affect of churn on DAT have been investigated. The network latency has been taken into consideration for measuring the performance of DAT. Table 1 shows the assumed random parameters of real-time latency[3] for each of the different network links in the system.

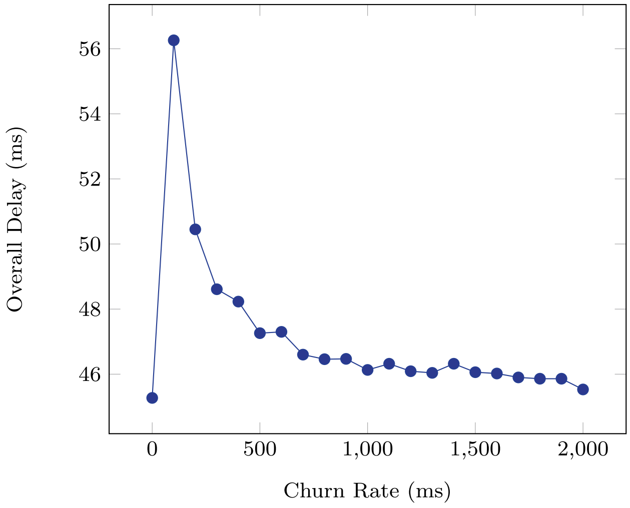

Fig. 7 Churn affect on DAT

We should call the reader’s attention to the fact that the IoT gateways are the peers in DAT and not the IoT devices. The members of $\mathcal{N}$ are not part of DAT itself and are connected through the peers in DAT (i.e. members of $\mathcal{W}$). Therefore, the lookup time represents the required time to get the address data from DAT through a peer in the overlay. The Kademlia implementation [4] of PeerSim simulator [25] has been used for the performance experiments. The implementation has been slightly modified to fit our proposed model. In the implementation and as with uTorrent[5], the popular implementation of Kademlia, system wide replication is set to 8 and the lookup parallelism is set to 4. The results of researches [15, 26] that focus on studying these two factors and other parameters in Kademlia [24] implementation to improve the lookup latency in DHT based implementations can be applied on DAT.

The frequent joining and leaving of nodes in p2p network, known as churn [12], might increase the lookup delay by requiring to connect to different nodes due to leaving of previously available nodes. To evaluate the efficiency of the DAT for handling issues of robustness, availability, and replication, we performed a set of experiments where in a network of 10,000 IoT gateways 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. During the evaluation, address lookup request of 100 requests per second have been issued. As shown

[3] https://wondernetwork.com/pings

[4] http://peersim.sourceforge.net/

[5] https://www.utorrent.com/

in the presented result in Fig. 7, there is 11 ms delay comparing to the network without churn in the address lookup time in DAT when the churn rate is 0.1 second (i.e. every 100 millisecond either an IoT gateway leaves or joins DAT) and less than one millisecond delay when the churn rate is higher than 1.6 second between each occurrence.

As part of the evaluation, the proposed model of address distribution using DAT has been compared with the models proposed in [19] and [21] in terms of latency, operational cost and privacy. Unlike DAT that creates a structured overlay for address distribution and end-to-end communication, the model proposed by Kavalionak et al. [19] uses the cloud computing technology in addition to a number of public proxies and the model proposed by Kfoury et al. [21] uses blockchain technology for address distribution and end-to-end communication.

We created a simulated network with 5,000 to 10,000 gateways. In our environment, there are 1000 IoT nodes that perform 1000 requests (i.e. a request per second) and have been distributed uniformly at random among the IoT gateways. Nodes in [19] have to choose proxy nodes in their proxy list to perform the NAT traversal approach through them. The proxy nodes can be chosen from the public nodes or the cloud proxy. Since the choice of public proxy nodes in this model requires a continuous pulse messages (i.e. keep-alive messages) which is not sufficient for the constrained IoT devices, here we consider the list of proxy nodes includes only the cloud proxy.

As it appears from Fig. 8 the lookup process of accessing an address of a NATed device in cloud based or blockchain based schemes are higher than the proposed model. But comparing to the cloud based and blockchain based models, the latency of our model can be increased logarithmically depending on the number of participating nodes in DAT. This issue has been discussed in more detail in Sect. 5.3.

In addition to latency, both rented cloud server and publishing in the blockchain require payment to the rented cloud provider or to the miners to execute the transactions within the blockchain. The prices of the cloud instances vary based on the providers, compute resources, instances, usage time, and so on[6]. The prices of publishing in the Ethereum blockchain used by [21] (e.g. about 0.0165 ETH for creating a mapping and 0.0053 ETH for registering a device) is based on Ethereum Gas price [7]. Therefore, the required operational cost are higher comparing to the DAT model that requires zero operational cost (i.e. payment for the nodes).

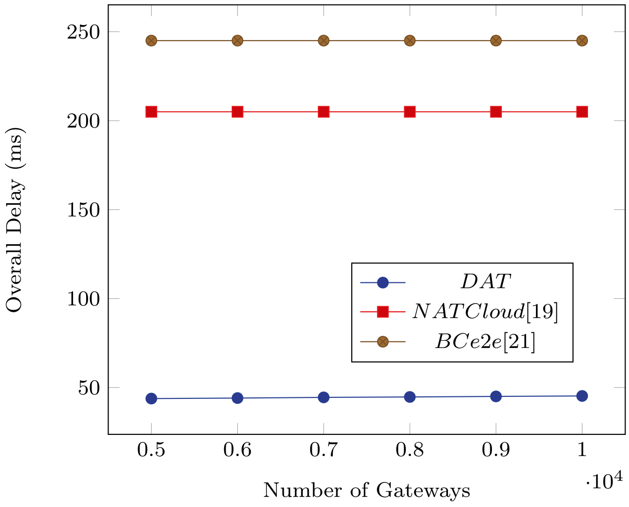

Fig. 8 Address lookup of the NATed devices

Table 2 Required operation time in a DAT quadruple by a microcontroller

Operation	Required time
XOR operation	8 ms
RNG	5 ms
SHA256 (Common ID and RN)	2 * 227 ms
AES-128-CBC encryption	288 ms
Add a quadruple in DAT	**805 ms**
XOR operation	8 ms
SHA256 (Common ID)	227 ms
AES-128-CBC decryption	348 ms
Get a quadruple from DAT	**583 ms**

In term of privacy, the address of nodes in [19] are known by the rendezvous servers and the cloud. While using blockchain technology in [21] removes the trusted third party and provides a mechanism to share the trust, but the addresses of nodes in this model are publicly available in the blockchain. The availability of addresses in public blockchain might cause serious security problems such as DoS of the nodes using their addresses. In contrast, the addresses in DAT are encrypted and can be accessed only by authorized nodes in the network (see Theorem 2 in Sect. 5.1 for the proof).

The IoT resources mostly have limited computation power. Therefore, the required computation of any proposed model has to be minimal and executable by those nodes. This issue is taken into consideration while developing DAT. Each IoT node generates the header and encrypts the address data. Therefore, it has to perform some cryptographic operations (i.e. symmetric encryption and hashing) during the process of quadruple generation in DAT. The quadruple generation process has been tested on an MCU with single-core 32-bit 80 MHz microcontroller. The SHA256 [9] 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[8] for the ESP8266 IoT devices has been used.

[6] https://www.simform.com/compute-pricing-comparison-aws-azure-googlecloud/

[7] https://ethereum.org/en/developers/docs/gas/

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

During the test, each of the cryptographic operations has been repeated 20 times, and their mean value is registered. The average time required by the microcontroller to perform the encryption, decryption and hashing to add or get a quadruple from DAT is shown in Table 2.

5.3 Computation Cost Analysis

Suppose that a node $i \in \mathcal{N}$ has a set of friend nodes $\mathcal{F}_i$. Let N_f represents the number of nodes in $\mathcal{F}_i$. Suppose that the ratio of nodes in $\mathcal{F}_i$ that has to be informed is β. Therefore, there are $\beta.N_f$ $AP2$s that have to be issued. Let's suppose that the cardinality of peers in the DAT overlay $\mathcal{W}$ is N_w. Suppose that γ is the ratio of uninformed nodes in $\mathcal{F}_i$ that will issue a *DRP* after receiving their corresponding quadruple from the DAT overlay. During the analysis, we focused on the number of $AP2$s that have to be issued and put the quadruples in the DAT overlay and the number of requests that have be sent on the other hand in order to get the stored quadruple from the DAT overlay. We discuss the cost of the protocol in two phases:

- Generating and storing a quadruple (C_s)
- Requesting and getting the quadruples (C_g)
- Removing the quadruples from the DAT overlay (C_r)

The storing of a quadruple is done by issuing an $AP2$ by the corresponding peer $w \in \mathcal{W}$ and is equal to $C_s = O(log(N_w^{\beta N_f}))$. The requesting and getting the stored quadruples will be issued βN_f times by the uninformed members of $\mathcal{F}_i$ and the cost is equal to $C_g = O(log(N_w^{\beta N_f}))$. The cost of removing the quadruples from the DAT overlay taking into consideration the ratio of γ is equal to $C_r = O(log(N_w^{\gamma\beta N_f}))$. The drawback of DAT, as it is clear from C_s, C_g, C_r and is slightly notable from Fig. 8 as well, is that when in the proposed model the number of fog nodes (i.e. gateways) increases the delay of the processes in DAT model increases logarithmically. The reason is the use of the DHT technology for store and lookup in which the lookup time among c peers is $log(c)$.

6 Conclusions

In this paper the Distributed Address Table (DAT), a decentralized, secure and lightweight address distribution model has been proposed. The DAT is based on the Distributed Hash Table (DHT) and can be integrated in the approaches that require distribution of addresses such as NAT traversal approaches. It satisfies the weak anonymity, address privacy and soundness properties that have been considered necessary for such design. During the design of the DAT model, the constrained nature of the IoT devices has been taken into consideration. Therefore, all the required cryptographic techniques have been chosen carefully and tested to be executable in the IoT devices with constrained resources. At the same time, this limitation should not reduce the required security by the model. The proposed model uses peer-to-peer scheme as its underlying communication to ensure that all participating devices are accessible at any given time. This is achieved through simple, yet secure and efficient decentralized model. The DAT model has been tested and analysed in terms of required computation time, overall delay, churn resilience, time complexity and security. It also has been compared with two relevant models of address distribution, namely the cloud based and blockchain based models. The results showed that the proposed DAT model is efficient in terms of latency, cost and privacy. In addition, the security properties of the proposed model have been proved.

Some open problems remain related to the proposed model. On one hand, the delay in DAT increases logarithmically as the number of nodes increases. This problem has to be addressed in future works. On the other hand, to distribute the addresses in DAT, a separate lookup in the overlay for each of the friend nodes is issued which will add a significant overhead to the system, there might be a future study to improve the efficiency of the address distribution process regardless of the number of friend nodes.

Acknowledgements 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 the European Union, co-financed by the European Social Fund. EFOP-3.6.2-16-2017-00013, 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, stipendium hungaricum program and by the Ministry of Science, Research and the Arts Baden-Württemberg Germany.

Funding Open Access funding enabled and organized by Projekt DEAL.

References

1. Becker N, Rizk A, Fidler M (2014) A measurement study on the application-level performance of lte. In 2014 IFIP Networking Conference 1–9. IEEE
2. Bhatnagar N (2019) Mathematical Principles of the Internet, Two Volume Set. CRC Press
3. Bormann C, Castellani AP, Shelby Z (2012) Coap: An application protocol for billions of tiny internet nodes. IEEE Internet Comput 16(2):62–67
4. Boucadair M, Penno R, Wing D (2013) Universal plug and play (upnp) internet gateway device-port control protocol interworking function (igd-pcp iwf). RFC 6970, RFC Editor
5. Chattopadhyay D, Samantaray A, Datta A (2018) Device microagent for iot home gateway: a lightweight plug-n-play architecture. ACM SIGBED Review 15(2):16–23
6. Cheshire S, Krochmal M (2013) Dns-based service discovery. RFC 6763, RFC Editor
7. Cheshire S, Krochmal M, Sekar K (2008) Nat port mapping protocol (nat-pmp). RFC 6886, RFC Editor
8. Cho SW (2015) P2p-based mobile social networks. In 2015 10th International Conference on P2P, Parallel, Grid, Cloud and Internet Computing (3PGCIC) 141–145. IEEE
9. Eastlake D, Hansen T (2006) Us secure hash algorithms (sha and hmac-sha). RFC 4634, RFC Editor
10. Egevang K, Francis P et al (1994) The ip network address translator (nat). RFC 1631, RFC Editor
11. Guo T, Shenoy P, Ramakrishnan K, Gopalakrishnan V (2018) Latency-aware virtual desktops optimization in distributed clouds. Multimedia Systems 24(1):73–94
12. Herrera O, Znati T (2007) Modeling churn in p2p networks. In 40th Annual Simulation Symposium (ANSS'07) 33–40. IEEE
13. Herry H, Band E, Perkins C, Singer J (2018) Peer-to-peer secure updates for heterogeneous edge devices. In NOMS 2018-2018 IEEE/IFIP Network Operations and Management Symposium 1–5. IEEE
14. Hunkeler U, Truong HL, Stanford-Clark A (2008) 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) 791–798. IEEE
15. Jimenez R, Osmani F, Knutsson B (2011) Sub-second lookups on a large-scale kademlia-based overlay. In 2011 IEEE International Conference on Peer-to-Peer Computing 82–91. IEEE
16. Johnston AB, Burnett DC (2012) WebRTC: APIs and RTCWEB protocols of the HTML5 real-time web. Digital Codex LLC
17. Kamel MBM, Crispo B, Ligeti P (2019) A decentralized and scalable model for resource discovery in iot network. In 2019 International Conference on Wireless and Mobile Computing, Networking and Communications (WiMob) 1–4. IEEE
18. Kamel MBM, Ligeti P, Reich C (2021) Lamred: Privacy preserving and location aware multi layer resource discovery for iot. Submitted to Acta Cybernetica
19. Kavalionak H, Payberah AH, Montresor A, Dowling J (2016) Natcloud: cloud-assisted nat-traversal service. In Proceedings of the 31st Annual ACM Symposium on Applied Computing 508–513. ACM
20. Kermarrec A-M, Pace A, Quema V, Schiavoni V (2009) Natresilient gossip peer sampling. In 2009 29th IEEE International Conference on Distributed Computing Systems 360–367. IEEE
21. Kfoury EF, Gomez J, Crichigno J, Bou-Harb E, Khoury D (2019) Decentralized distribution of pcp mappings over blockchain for end-to-end secure direct communications. IEEE Access 7:110159–110173
22. Lyu Z, Tu X, Pei X, Liang H, Sarem M (2017) Netlet: A simple and versatile network middleware for peer-to-peer application program. Comput Electr Eng 59:1–14
23. Mamdouh M, Elrukhsi MA, Khattab A (2018) Securing the internet of things and wireless sensor networks via machine learning: A survey. In 2018 International Conference on Computer and Applications (ICCA) 215–218. IEEE
24. Maymounkov P, Mazieres D (2002) Kademlia: A peer-to-peer information system based on the xor metric. In International Workshop on Peer-to-Peer Systems 53–65. Springer
25. Montresor A, Jelasity M (2009) PeerSim: A scalable P2P simulator. In Proc. of the 9th Int. Conference on Peer-to-Peer (P2P'09) 99–100, Seattle, WA
26. Roos S, Salah H, Strufe T (2017) On the routing of kademlia-type systems. Advances in Computer Communications and Networks
27. Rosenberg J (2008) Traversal using relays around nat (turn): Relay extensions to session traversal utilities for nat (stun. RFC 5766, RFC Editor
28. Rosenberg J (2010) Interactive connectivity establishment (ice): A protocol for network address translator (nat) traversal for offer/answer protocols. RFC 5245, RFC Editor
29. Srirama SN, Liyanage M (2014) Tcp hole punching approach to address devices in mobile networks. In 2014 International Conference on Future Internet of Things and Cloud 90–97. IEEE
30. Stephenson A, Namiot D (2015) On data transfer between mobile web clients. International Journal of Open Information Technologies 3(1):30–40
31. Tian L (2012) Lightweight m2m (oma lwm2m). OMA device management working group (OMA DM WG), Open Mobile Alliance (OMA)
32. Tsirtsis G, Srisuresh P (2000) Network address translation-protocol translation (nat-pt). RFC 2766, RFC Editor
33. Vaquero LM, Rodero-Merino L (2014) Finding your way in the fog: Towards a comprehensive definition of fog computing. ACM SIGCOMM Computer Communication Review 44(5):27–32
34. Wing D, Cheshire S, Boucadair M, Penno R, Selkirk P (2013) Port control protocol (pcp). RFC 6887, RFC Editor
35. Wing D, Matthews P, Mahy R, Rosenberg J (2008) Session traversal utilities for nat (stun). RFC 5389, RFC Editor
36. Zhang H, Wen Y, Xie H, Yu N (2013) A survey on distributed hash table (dht): Theory, platforms, and applications. survey paper

Publisher's Note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.

Mohammed B. M. Kamel obtained his Master in Computer Science from the University of Baghdad and IT Adminstration from Technischen Universitat Berlin. He got the Minister of higher educations and deputy of Prime ministers of Iraq awards for holding the highest GPA in undergraduate program. Currently he is a PhD researcher in Eotvos Lorand University (ELTE) and Hochschule Furtwangen University (HFU), a member of the Institute of Data Science, Cloud Computing and IT-Security. Recently, he got the Gold award from EIT Health Innovation day of having the best innovative project which later has been presented in EIT WinnerDay in Paris. His main research interests are in the area of network security and applied cryptography and mainly focus on security and privacy in distributed environments.

Peter Ligeti is an associate professor at Eotvos Lorand University of Sciences Faculty of Informatics. Earlier he was a part time researcher at Alfred Renyi Institute of Mathematics. His main research interests are cryptography (secret sharing, secure distributed protocols, network coding) and combinatorics (combinatorial optimization, combinatorics on words).

Adam Nagy is an assistant lecturer and PhD candidate at Eotvos Lorand University of Sciences, Faculty of Informatics. His main research interests are applied cryptography (location based cryptography, protocols, implementation) and high performance computing (parallelization, algorithms from number theory).

Christoph Reich is professor (since 2002) at the faculty of computer science at the university of applied science in Furtwangen (HFU) and teaches in the field of network technologies, IT protocols, IT security, Cloud Computing, and Machine Learning. He is CIO of the HFU Information- and Media Centre, that is the scientific director for the IT data centre, Online-Services, Learning-Services, and library department. He is head of the Institute of Data Science, Cloud Computing and IT-Security (IDACUS; idacus.hs-furtwangen.de). Several funded research projects (FP7 EU, BMBF, MWK) have been accomplished successfully.