IoT Security via Address Shuffling: the Easy Way

Francesca Nizzi, *Student Member, IEEE,* Tommaso Pecorella, *Senior Member, IEEE,*
Flavio Esposito, *Member, IEEE,* Laura Pierucci, *Senior Member, IEEE,* and Romano Fantacci, *Fellow, IEEE*

***Abstract*—Securing Internet of Things (IoT) devices and protecting their applications from privacy leaks is a challenge, due to their weak (computational and storage) capabilities, and their proximity with sensitive data. Considering the resource-constrains of such devices, their long lifetime, and the intermittent connections, classical security approaches are often too difficult or impractical to apply. Moving Target Defense is an established technique whose goal is to lower the attack surface to malicious users by constantly modifying device footprint. Changing the address to an IoT device without privacy leaks is, however, a non-trivial task. In this paper, we propose a novel method to perform a network-wide (IP and MAC) address shuffling procedure, called Address Shuffling Algorithm with HMAC (AShA), which is simple to implement, and whose network overhead is minimal. To demonstrate its effectiveness, we analyze our approach via theoretical analysis and simulations. Our analysis shows how AShA parameters can be adapted to various network sizes while our simulations results show how AShA can be used to successfully perform a global collision-free address renewal on networks of more than 2000 nodes using 16-bit addresses.**

***Index Terms*—Internet of Things, Address shuffling, Moving Target Defense, Security.**

I. Introduction

Despite the rapid increase of both Internet of Things (IoT) and Wireless Sensor Network (WSN) markets [1], security and privacy concerns could severely impact their massive deployments and applications. Smart cities [2], smart campuses, disaster response [3], industrial, smart home and healthcare, are merely a few examples of sectors that could benefit from, and be impacted by, a lack of private and secure IoT communication. Being small, low-power, Internet-connected, and capable of observing or modifying the physical world, IoT devices are a honeypot for attackers [4]. As a consequence, securing IoT devices is (arguably) more important than securing 'classic' Internet hosts. A lack of IoT security can result in a range of consequences, from privacy losses to physical harm to the users [5], [6]. Another peculiarity of the IoT devices is the need to be both energy and bandwidth efficient. This requirement comes from the need to prolong the node life (most devices are battery-operated) and to allow ultra-dense node deployments.

To avoid unnecessary data transmission, payloads are often compressed, and IoT protocols have shorter headers compared to the Internet counterparts. For example, the IPv6 header is often compressed using RObust Header Compression (ROHC) [7] or IPv6 over Low power Wireless Personal Area Network (6LoWPAN) [8]. Header compression, along with IPv6 Stateless Address Autoconfiguration (SLAAC) [9], is however a serious privacy threat: an attacker may discover the IP address of the nodes by merely analyzing their Medium Access Control (MAC) address, opening its way to analysis of node capabilities and vulnerabilities. Moreover, passive attackers can infer network topologies and learn what are the nodes functionalities even without compromising any system (e.g., they could infer which nodes are responsible for perimeter surveillance planning for another attack) [10].

To reduce the attack surface within IoT network and system security, a viable technique is to introduce randomness in the network behavior. The efforts of some attackers are diminished or even vanished without the ability to gather or predict enough information before the network parameters change. Examples of security techniques that leverage randomness to secure a communication are encryption key refresh and application-level behavior variations. Random (application) protocol behavior can be implemented, for example, by modifying the periodicity of a node data report, or by adding random data to regular packet payloads. Despite being a valuable defense mechanism approach, randomizing the application behavior alone is insufficient to prevent routing or other types of attacks generated by guessing the *physical* node placement in the network: an attacker can still use the MAC addresses to learn the network topology. To add a MAC-level randomness, a system could periodically change the MAC address of each node. This technique alone has, however, various drawbacks. In particular, the signaling overhead required to coordinate the address change is significant. While in wired network the overhead may not be a concern, when devices are constrained by bandwidth and power, such overhead may severely impact the IoT system performance. It is hence important to devise (MAC) address renewal methods with minimal impact on the network signaling overhead.

To this aim, in this paper we present an a novel address shuffling technique that we call *AShA*, as in **A**ddress **Sh**uffling **A**lgorithm. *AShA* is energy-efficient, has minimal impact on the network overhead, and it is easy to implement. The key novelty behind *AShA* is a cryptographic hash that enables a controlled and collision-free address shuffling. Only the legitimate nodes and the network controller are able to predict the address renew outcomes, and from the point of view of the attacker, the addresses follow a random pattern over time.

We evaluate the efficiency of our proposed method with respect to the number of nodes in the network both theoretically and through simulations. Our results show that, for typical network sizes (i.e., less than 700 nodes for each Personal

F. Nizzi, T. Pecorella, L. Pierucci, and R. Fantacci are with the Dpt. of Information Engineering, University of Florence, Via di S. Marta, 3 - 50139 Florence, Italy, e-mail: name.surname@unifi.it

F. Esposito is with the Computer Science Department, Saint Louis University, MO, 63103 USA, e-mail: flavio.esposito@slu.edu

Area Network (PAN)), the address renew procedure does not add any overhead, while for larger PANs (i.e., up to 2300 nodes) the overhead is tunable, and it depends on network management decisions.

The rest of the paper is structured as follows. In Section II we detail the considered scenarios and threat models, while Section III presents related solutions on privacy and 6LoWPAN available in literature. Section IV describes the proposed method used to change addresses, along with our theoretical analysis. In section V we show a practical use-case of AShA when applied to an Institute of Electrical and Electronic Engineers (IEEE) 802.15.4 network. Finally, in Section VI, we show how analytical and simulations results match and we draw guidelines for a general use of the proposed method. In Section VII we illustrate the conclusions and we remark the main benefits of the proposed system.

II. Threat Model and Motivating Scenarios

In this section we describe our attacker models and later on we motivate our low-overhead randomized addressing scheme with a few scenarios and applications.

Debates regarding what are the major security threats in Ad-Hoc Networks with an Internet Protocol (IP) address auto-configuration have been floated before (see, $e.g.$, [11]). We have identified four major security threats:

1) *Address Spoofing Attack*. An attacker may spoof IP addresses and inject (replayed or forged) packets.
2) *False Address Conflict Attack*. An attacker may leverage address collision prevention mechanisms to perform a denial of service.
3) *Address Exhaustion Attack*. An attacker may exhaust the space of available IP addressing by declaring usage of a large number of them, preventing new devices to to join the network for lack of available IP addresses.
4) *Negative Reply Attack*. An attacker may continually transmits a (false) deny message when a new node tries to obtain a new address. In this case, the attacker maliciously acts as a Network Coordinator.

Beside address spoofing, all attacks are based on vulnerabilities of the link-local protocols, and in particular, Duplicate Address Detection (DAD) and Neighbor Discovery Protocol (NDP). Message encryption and validation at layer 2 is normally used to mitigate such attacks.

Preserving Privacy in Internet of Medical Things. Preserving privacy of medical records is often a law requirement [12]. Aside from the potential benefits of protecting electronic medical records, which include lab tests, images, diagnoses, prescriptions and medical histories, recently body area networks have been used for therapeutics monitoring, for example of pacemakers. With such technology, healthcare providers may instantly access patient histories that are relevant to future care and patients can take ownership of their medical records. In general, medical record as well as IoT devices offer the potential for greater privacy and better access to patient data when they are needed. Attackers may, however, leverage these information for data leaks or even to hijack pacemakers or other body area devices via address spoofing or address exhaustion attacks[1].

Preserving Privacy in Edge Computing. Edge computing is a paradigm based on outsourcing computational tasks from a distributed set of end-hosts, or viceversa, edge nodes can be used to release pressure from data centers [13]. In edge computing, privacy preservation is more challenging since end-nodes may collect sensitive information such as identity or locations of end users (that may contain sensitive information otherwise stored at the network core). Moreover, since edge end-hosts are scattered in large areas, centralized control is becoming difficult. A poorly secured distributed edge computing network can be the entry point for intruders. Once inside the network, the intruder can mine, steal, damage or leak users or machine sensitive data. Location privacy is arguably one of the most important models for privacy, since pieces of hardware can be linked to owners (data). Since in edge computing, offloaded tasks may include location, trajectory and even mobility habits, such information revealed to an adversary may be part of a weaponization phase and may hence lead to subsequent more severe attacks.

Securing Industrial Control Systems and Smart Grids. The integration of IoT into industrial systems has been fostered to catalyze the potential automation of tasks such as manufacturing or product tracking. These systems often rely on strict timings and precise execution, requiring close monitoring to function safely and effectively. Control systems like these have long been used, but recently, several sensors have been attached to the Internet for remote maintenance shifting to IoT architectures. An example of that are power grids, that are quickly becoming smart grids, using automated metering infrastructure to remotely sample usage data from residential meters. A simple address spoofing attack may potentially jeopardize an entire production chain with costly consequences. As shown in [14] smart meters readings can disclose information about the time that the house is empty (i.e., it is safe to break in) or even the TV programs that a user prefers to watch.

Protecting Injured or Lost People in a Disaster Scenario. During or after a disaster scenario, such as hurricane Katrina, mobile ad-hoc networks are established by first responders to cope with the lack of (disruptive) network infrastructure; attackers may intercept network traffic among first responders to arrive first on victims and identify subjects for human or organ traffic; these attacks may be delivered for example, via a False Address Conflict Attack.

III. Related Work

In literature, several solutions for IP address auto-configuration have been proposed. The authors in [15] classify these solutions in three major groups: stateful, stateless and hybrid. In *stateful* address auto-configuration approaches, all nodes consult a logically centralized node or a set of distributed agents to obtain a valid IP address. The Dynamic Host Configuration Protocol (DHCP) protocol [16] is part of this family. In *stateless* address auto-configuration approaches,

[1]https://www.theguardian.com/technology/2017/aug/31/hacking-risk-recall-pacemakers-patient-death-fears-fda-firmware-update

every joining node self-assigns an address; such address may be chosen at random or with a predetermined algorithm. Stateless address auto-configuration approaches lead to addresses conflicts, that are resolved with overhead, delays or with a centralized coordinator and, as consequence, they scale poorly. Without a coordinator, two nodes may end up choosing the same address. To avoid such address uniqueness issue, the DAD [17] algorithm is used. Every node, after selecting an address, performs DAD to detect any collision. SLAAC [9] is part of this family. Our solution enables to auto-configure addresses without the need of explicit address duplicate checking, as this function is already fulfilled by the network coordinator. However, unlike DHCP, our method does not require any message exchange between the node coordinator and the nodes, with the exception of a network-wide address renew message broadcasted from the network coordinator to all the nodes.

It may be unsuitable to use the DHCP protocol in an ad-hoc network, due to the limited device's energy, its (possible) mobility, and the insufficiently resilient infrastructure (a dedicated server may become unavailable). Existing works have proposed two approaches in response to the drawbacks of DHCP: either an adjustment of DHCP to enable a dynamic IP address allocation, or the use of a coordinator that assigns addresses. For example, in [18], authors assign an IP address to the mobile devices using an Ad-Hoc DHCP scheme, while in [19] a central node (called ZigBee Coordinator) is responsible for assigning addresses.

In [20], the authors propose a dynamic host configuration to assign the IP addresses based on a distributed agreement problem: when a new node joins the network, a selected node proposes a candidate IP address: if such proposal reaches consensus, the proposed address is assigned to the new node, otherwise another address is proposed (a given number of times).

The Internet-Draft on *Ad Hoc Address Auto-configuration* [21] proposes a stateless approach to auto-configure IP addresses that are unique in the network. A DAD-like protocol is used to check the proposed address uniqueness before the auto-assignment. Briefly, a node chooses two addresses: a temporary used as IP sender address only, and a proposed sent to the other nodes to check if it is unique in the network. If no answer is received within a given interval, the node assigns the proposed address to itself. Obviously, if an answer is received, the node changes the proposed address and tries again.

In [22], an hybrid scheme is used: the auto-configuration is performed by the device but, to maintain the information in the network and to detect duplicate addresses, an elected coordinator (called Address Authority) is used.

These solutions focus only on the IP address auto-configuration and do no consider several critical points, such as the static nature of MAC address/Interface Identifier (IID). In [23], the author studies some problems arising from a static IID. The author suggests that not only the IP address must be periodically changed, but also the layer 2 address (MAC address) must be changed as well.

The authors in [24] provided a method based on an extension of IPv6 over Low power Wireless Personal Area Network - Neighbor Discovery (6LoWPAN-ND) to change both L2 and L3 addresses, while maintaining the header compression (as well as all other functionalities of the 6LoWPAN) and a small routing table. The technique presented in this paper, with respect to [24], maintains all the original benefits, reducing the network overhead to zero for small to medium size networks. Moreover, the active sessions and routing path management during the address renew are greatly simplified.

IV. Proposed Method

Existing address shuffling techniques are based on complex protocols [25], [26]. All of them assume that a central entity (a coordinator) knows about all the devices associated to the network, and it can manage reliably the association/disassociation of each node in the network. This assumption is usually true for all the practical IoT networks. In this paper we will assume that the association/disassociation phases are managed at MAC level.

As seen in Section III, the coordinator usually (re)assigns the node addresses and ensures a lack of address collision. A simple way to re-assign an address to a node is to let the coordinator choose it randomly, avoiding address collisions. However, the coordinator must send a unicast message to every device in the network, incurring in huge signaling costs. Another solution is to let each node choose its new address randomly, and to let the coordinator the duty to correct collisions. To do so, however, the coordinator is required to send a unicast message to inform every device in the network about its correction decision. This solution is costly (the signaling cost is very high), and, especially when applied to medium or large networks, often requires extensive convergence time. An alternative is to let the node itself choose the address randomly, using a DAD (or an equivalent protocol); this solution is complex, also requires extensive signaling, and may have similar execution times. Moreover, in a multi-hop ad-hoc networks, multicast messages might not be supported, requiring a fallback to even less efficient unicast communication.

We propose a novel algorithm for address auto-configuration in which each node autonomously calculates the new address, and such address is known to the network coordinator in advance. The coordinator will only signal to all nodes that the address needs to be changed, while avoiding potential conflicts. The idea is to allow each node to choose autonomously a new (MAC-IPv6) address (pair), guaranteeing at the same time that each new address is unique in the network. In our proposed scheme, the PAN coordinator role is to choose a set of parameters that, once spread to the nodes, will trigger a collision-free address renew.

The goals of an optimal ad-hoc network addresses auto-configuration algorithm are: *dynamic address configuration*, *uniqueness*, *robustness* and *scalability* [27]. With such auto-configuration algorithm, a node is able to obtain dynamically a unique IP address without static or manual configuration; moreover, the algorithm must be able to adapt as the network conditions evolve and without performance degradation as

the number of devices grows. We show how our proposed algorithm meets all the above requirements.

A. AShA

In this section we present the details of our approach, named Address Shuffling Algorithm with HMAC (AShA).

AShA is based on a clever use of hash functions, and in particular Keyed-Hash Message Authentication Code (HMAC). We begin by recalling the definition of HMAC, and then we describe how it is used within our address shuffling technique. Eq. 1 describes the HMAC function as defined in [28]:

$$\text{HMAC}(K, m) = H\big((K' \oplus \text{opad}) \parallel H((K' \oplus \text{ipad}) \parallel m)\big) \quad (1)$$

where:

- $H(\cdot)$ is a cryptographic hash function that operates on blocks of B bytes iteratively,
- m is the message to be hashed,
- K is an arbitrary secret key.
- K' is a B-bytes key derived from K,
- ipad is the byte `0x36` repeated B times,
- opad is the byte `0x5C` repeated B times.

The properties of the HMAC function greatly depends on the hash function being used. However, we can expect that the hash function fulfills the three hash functions properties, i.e., pre-image resistance, second pre-image resistance, and collision resistance. An interesting consequence of the collision resistance is that the hash function should be able to map the domain in the whole codomain, that is, all output bits are used.

Thanks to the hash properties, it is possible to derive a pseudo-random (IPv6, MAC or both) address of node i from two parameters:

$$\text{ADDR}_i(K) = \text{HMAC}(K, \text{ID}_i) \quad (2)$$

where ADDR_i is the new address of node i, ID_i is an identifier for node i (e.g., its serial number), and K is a secret key. Note that it is possible (and advisable) to have a per-node key K_i. Without loss of generality, in the rest of the paper we will assume a shared key K for all the nodes of the network.

This method can be used by the nodes to self-assign a pseudo-random MAC/IPv6 address. As shown by the Birthday paradox, address collisions are improbable, but still possible, and the collision probability is given by [29]:

$$P(n, d) = 1 - \frac{d!}{(d-n)!d^n} \approx 1 - \exp\left(\frac{-n(n-1)}{2d}\right) \quad (3)$$

where d is the hash function image size (i.e., the address length) and n is the number of nodes. As a consequence, an address collision resolution mechanism is needed (see also [10]).

As an example, in Fig. 1 the birthday paradox effects for a 16-bit number (2^{16} available addresses) are shown. In other terms, using a simple HMAC (or any random generated number) to decide a new address would lead to higher collision probability with a network of merely 200 nodes.

This mechanism is impractical for periodic address changes since the node identifier and the key are constant. Even though a key renewal mechanism is in place, the key validity time should be considerably longer than the address renewal time, simply because the key renewal protocols generates significant overhead, which is against our design goals.

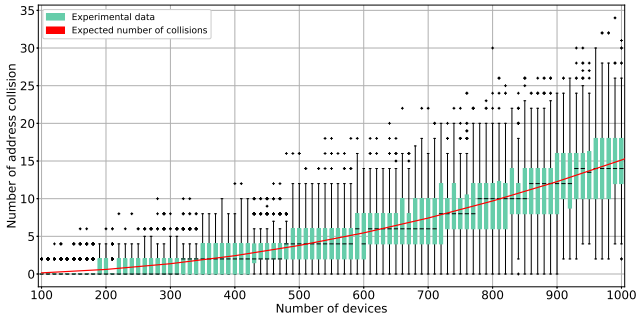

Fig. 1. Birthday Paradox with 2^{16} MAC-16 addresses.

To allow a *periodic* and *overhead controlled* address change, the HMAC function must be changed by adding one more parameter, which can be set by the network coordinator requesting the address change action. By carefully choosing the new parameter value, the network coordinator can also avoid address collisions, as we demonstrate in Sec. VI. In the proposed system, a new address is created from a *triplet*:

$$\text{ADDR}_i(K, r) = \text{HMAC}\left(K, (\text{ID}_i \parallel r)\right) \quad (4)$$

where r is an address refresh index (named in the following METAindex) managed by the network coordinator. With this method, the secret key can be updated independently from the address refresh procedure, and the network coordinator can control the address collisions by not using the values of r that would cause one, i.e., the coordinator changes r when a collision is detected. The secret K should be changed periodically in order to prevent possible attacks. As a matter of fact, it should be avoided to use more than once the same METAindex r without changing the secret key K. Toward this end, the coordinator must use a suitable key distribution protocol to refresh K.

B. The AShA algorithm

Our proposed algorithm, called AShA, is previously run by the network coordinator and, at a later time, by the devices. The key K and the node ID are known by the device, while the METAindex parameter (r) is provided by the network coordinator. As a consequence, any node can calculate its MAC and IPv6 address independently.

When an address shuffling is required, for example as part of a moving target defense strategy, the network coordinator sets r to a particular value and evaluates, for a given K, all new addresses of the devices associated to its network. If a collision is found, the coordinator changes the METAindex and re-calculates all addresses. The loop ends when the network coordinator does not find collisions and subsequently, a multicast message with the r value to be used is sent to all devices. Algorithm 1 shows the pseudo-code of the network coordinator AShA algorithm. Parameters (K, r) are omitted for brevity.

Algorithm 1 network coordinator AShA

Init: $r \leftarrow \text{METAindex}_{\text{Cur}} + 1$
while $r \leq \text{METAindex}_{\max}$ **do**
  for all nodes **do**
    $\text{ADDR}_i \leftarrow \text{HMAC}\left(K, (\text{ID}_i \parallel r)\right)$
  end for
  Collisions $\leftarrow$ **False**
  if $\exists\, i, j \in$ nodes : $\text{ADDR}_i = \text{ADDR}_j$ **then**
    Collisions $\leftarrow$ **True**
  end if
  if Collisions $=$ **False then**
    $\text{METAindex}_{\text{Cur}} \leftarrow r$
    Send r
    Break
  else
    $r \leftarrow r + 1$
  end if
end while
if $r = \text{METAindex}_{\max}$ **then**
  Generate and distribute new key
  $r \leftarrow 0$
  Go to 2
end if

When the devices in the network receive the *address change multicast message*, with the new r value, they automatically evaluate their new MAC and IPv6 addresses.

This preventive control of the address collisions by the network coordinator, in addition to enable a simultaneous addresses shuffling in the network, avoids the use of onerous collision detection mechanism (e.g., DAD) used in IP networks. When the address reconfiguration is performed, the node is sure that its new (IPv6 and MAC) address does not collide with any other address in the network.

Moreover, AShA is able to mitigate all the security threats presented in Sec. II. In particular, address spoofing is automatically resolved once the network addresses are shuffled, false address conflict is not anymore possible because addresses are handled by the coordinator, and address exhaustion can be mitigated by jointly using address shuffling and per-device authentication. Negative reply attack is mitigated by AShA for the nodes already in the network, while for new nodes trying to join the network it is necessary to use MAC-level encryption.

C. Theoretical Analysis

In this section we analyze the AShA mathematical properties, and how to leverage the METAindex to further obfuscate the network.

Since AShA is based on a HMAC function, we expect that the probability of not having any collision in the network is based on the number of the nodes in the network. Once all the range of METAindexes has been used, it is necessary to renew the secret key K.

In order to avoid a frequent key renew process, it is advisable to be able to have a large enough pool of METAindexes to choose from. However, an attacker could discover the approximate network size by checking how many METAindexes are skipped and/or by counting the number of unicast address change messages. This potential information leakage could be used to perform further attacks, and it should be prevented.

An easy, and yet very effective way to further obfuscate the network operations is to split the METAindex in two parts: the Primary Index and the Secondary Index. The Primary Index is used to trigger the address change procedure, while the Secondary Index is used to find the right METAindex that does not generate a collision. In other terms, for each address shuffling the Primary Index is incremented, and the Secondary Index can be chosen randomly among the numbers that does not generate a collision. In this way, an attacker will not be able to analyze the AShA procedure to infer the network size.

The Primary Index and Secondary Index effects on AShA can be analyzed mathematically. Eq. (3) shows the classical Collision Probability formula. It can be demonstrated that, if the hash function follows the necessary hash properties (i.e., pre-image resistance, second pre-image resistance, and collision resistance), then eq. (3) can be applied also to the AShA case.

We define $P_H(n, d)$ as the probability that Primary Index r does not generate a collision using $\text{HMAC}\left(K, (\text{ID}_i \parallel r)\right)$, and $\widehat{P}_H(n, d, k)$ as the probability that exist at least one Secondary Index r' such as a Primary Index r does not generate a collision using $\text{HMAC}\left(K, (\text{ID}_i \parallel r \parallel r')\right)$, where n is the number of addresses generated, d the length of the address being generated, and k is the length of r' (Secondary Index).

It is possible to show that:

$$P_H(n, d) = \frac{d!}{(d-n)!d^n} \approx \exp\left(\frac{-n(n-1)}{2d}\right) \tag{5}$$

$$\widehat{P}_H(n, d, k) = 1 - \left(1 - P_H(n, d)\right)^k \tag{6}$$

The effect of the two Indexes is shown in Fig. 2, where is clear how the introduction of a Secondary Index obfuscates the AShA dependency on the network size. It must be stressed that, thanks to the Secondary Index, a Primary Index can be used multiple times, while without the Secondary Index, each Primary Index can be used only once before renewing the key K.

The overall algorithm complexity for each secret key is $O(n \times 2^k)$ where n is the number of devices, k is the METAindex size in bits, $O(n)$ is complexity of the hash and the collision search, and $O(2^k)$ is the complexity of the while loop. However, this value is spread among multiple METAindexes. The complexity needed to find a new viable METAindex depends on eq. (5).

V. Use Case: Applying AShA to LR-WPAN

A Low-Power and Lossy Network (LLN) is a type of network characterized by low energy consumptions, low bit-rate, and highly unreliable communication links. A Low-Rate Wireless Personal Area Network (LR-WPAN) is a special kind of LLN, and it is often based on IEEE 802.15.4 [30] physical and MAC layers. Its typical protocol suite is shown in Fig. 3. In this section, we discuss how we may apply our address shuffling proposal to LR-WPAN.

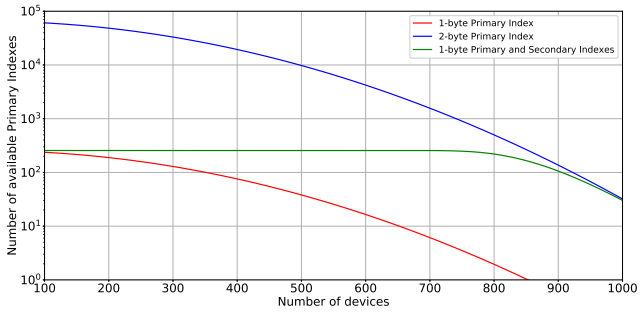

Fig. 2. Effects of Primary and Secondary Indexes on *AShA*.

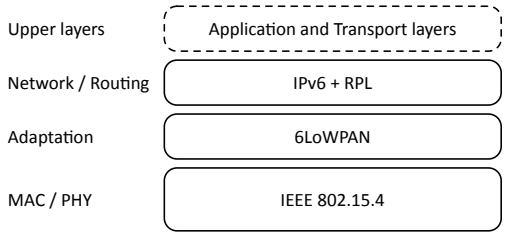

Fig. 3. LR-WPAN reference layers used in the AShA use-case.

Multi-hop WSNs are often organized in tree-like structures as, for example, in IEEE 802.15.4 Cluster-Tree PAN, RPL, etc. In such network topologies there are a *root* node, which corresponds to the PAN coordinator, all the other nodes communicate with the root by using multi-hop communications. A node, except for the root, has one (or more) *parent(s)*. If a parent node is reconfiguring, it blocks the communications of all the nodes using that node as a parent. As a consequence, there is a delay in the reconfiguration process and a possible energy waste.

The address reconfiguration approach suggested in [24] is to start the reconfiguration from the tree leafs. However, this solution requires an exact topology knowledge and an additional coordination among the nodes.

Our solution completely eliminates the need for a complex exchange of messages with the network coordinator: they can switch to the new address autonomously as soon as they receive the address renew message.

A. Involved technologies

In the following subsection we outline the main protocols used in our scenario.

1) IEEE 802.15.4: The IEEE 802.15.4 network can have different configurations, i.e., tree, multi-tree, and mesh. In all of them, a PAN coordinator is responsible for starting and maintaining the network, manage the node associations, and forward the PAN traffic to the Internet. Each network interface can use two different address types: Long (64 bits) and Short (16 bits). The Long address is globally unique, and it is assigned to the device by the manufacturer. The Short address (henceforth called MAC-16) is unique in the PAN, and it is assigned to the device by the PAN Coordinator during the association phase. A device can use either the Long or the Short address within the PAN, but Long addresses are discouraged due to their excessive length.

Theoretically, the available MAC-16 address space is 2^{16}. However, the following addresses are reserved [31]:

- `FF:FE` - Temporary address for devices associated to PAN but without a short address;
- `FF:FF` - Device not yet associated to PAN / Broadcast MAC-16 address;
- `80:00` - `9F:FF` - Multicast MAC-16 addresses.

As a consequence, only $2^{16} - 2 - 2^{13} = 57342$ addresses are available. To keep existing connections alive, it is useful to split the address space into two sets: one used before the addressing renew, and one after. In this manner, the PAN coordinator can keep forwarding, for a given period of time, the packets to the nodes that have active connections, even when their actual address is changed. Upon an address change, applications will have to re-negotiate a new connection with their endpoints.

For the sake of simplicity, we can use the Least Significant Bit (LSB) for this purpose but any bit is a good choice. We must note that this bit does not add anything to the security or secrecy of the network, as it is fairly easy for an attacker to discover the used bit by controlling the network address pattern before and after an addressing renewal operation. Summarizing, each node can choose one of the 28671 available addresses at each change of address.

Each node knows its own MAC-64, which is globally unique by definition. Moreover, we assume that all the nodes share with the PAN coordinator a Secret key. The Secret key could be installed during the network setup by using a secure network parameters installation method (see for example [32]). Key derivation and management are standard procedure and hence are out of the scope of this paper. However, a good approach could be to use the HKDF Scheme [33], [34].

2) 6LoWPAN and the neighbor discovery optimization: As mentioned in [35], the use of IPv6 technology can provide several benefits: from use of diagnostic and management IP-network tools, to the possibility to directly connect with other IP-based networks without using third-party components such as proxies or gateways. One of the main problems of using IPv6 as network protocol for LR-WPANs is the maximum physical layer packet size: in IEEE 802.15.4 this value is set to 127 bytes, that is, below the minimum IPv6 packet size (1280 bytes). Therefore, an adaptation layer between MAC and network layer is required.

6LoWPAN [31] mainly deals with packets compression, fragmentation and reassembly. The compression is performed by removing any fields in the IP, UDP, and ICMP protocols that can be obtained (or calculated) from the context, i.e., the MAC-layer header. A key factor of 6LoWPAN header compression is its efficiency with respect to the Internet Protocol version 6 (IPv6) address used. If the IPv6 address is based on the MAC address, then the compression is extremely efficient. Otherwise, its effectiveness is greatly reduced. As a consequence, it is mandatory to keep the two addresses paired and synchronized.

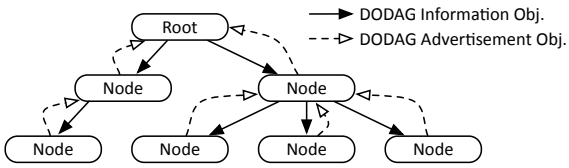

Fig. 4. Upward and Downward routes construction process. DIO messages are multicast while DAO messages are unicast.

For what concerns the Neighbor Discovery (ND) optimization, 6LoWPAN-ND [36] defines, among other things, optimized methods to adapt IPv6 ND [37] to the LR-WPAN context. In particular, the address collision detection task is assigned to the 6LoWPAN Border Router (6LBR). All nodes register to the 6LBR, and all ND and DAD checks (usually via multicast messages) are performed through unicast queries to the 6LBR. Even though not explicitly required by the standard, the 6LBR and the PAN coordinator functions are usually implemented within the same node.

3) LLN routing and RPL: LLNs topology is usually mesh-based. Moreover, thanks to the unreliable links and the long sleeping periods of the nodes, standard mesh routing protocols are not suitable. The Routing Protocol for Low-Power and Lossy Networks (RPL) [38] is a routing protocol specifically developed for a LLN. It is based on the construction and maintenance of a Destination-Oriented Directed Acyclic Graph (DODAG): a Directed Acyclic Graph (DAG) originating from a root node.

RPL defines three values to build and maintain a topology:

- *RPL Instance ID*: identifies a set of one or more DODAGs. At most, a node can be part of one DODAG in a RPL Instance but can belong to multiple RPL Instances.
- *DODAG ID*: is the identifier of a DODAG root and corresponds to a routable IPv6 address. The set of (RPL Instance ID, DODAG ID) identifies a single DODAG in the network.
- *DODAG Version Number*: is a value that, when changed by the root, triggers a complete DODAG refresh.

Thus, a DODAG is uniquely identified by the tuple {*RPL Instance ID, DODAG ID, DODAG Version Number*}.

RPL defines two types of routes: *Upward* (from the nodes to the root), and *Downward* (from the root to the nodes), as shown in Fig. 4. DODAG construction, maintenance, and Upward routes management are performed through DIO messages. This type of message is periodically sent by the DODAG root and by all the Full-Function Device (FFD) nodes in the network. Although DIO messages are sent using multicast, internal RPL mechanisms ensure that all nodes in the network will receive and process them correctly. Downward routes are propagated through DAO messages, sent by each node to the root, via any intermediate node. Upward route is mandatory while Downward route is optional, therefore DIO messages are always sent.

We will take advantage of the *DODAG Version Number* to implement the AShA algorithm in a LR-WPAN: when a *DODAG Version Number* changes, any node in the network is forced to use the new value. As a consequence, we can use the *DODAG Version Number* as a reliable mean to spread the address renew data and to trigger an address shuffling.

B. AShA implementation on LR-WPAN

The birthday paradox must be carefully evaluated when using AShA to calculate short (MAC-16) and IPv6 addresses. In general, for HMAC, the first collision is expected after evaluating approximately $\sqrt{\frac{\pi}{2}2^n} \approxeq 1.25 \cdot 2^{\frac{n}{2}}$ hashes, with n being the hash length. As an example, random or pseudo-random address generation makes sense in case of IPv6 addresses, where the full host part (64 bits) is used: the probability of having a collision becomes non-negligible only with 5.38×10^9 addresses actively used in the network. However, when the IPv6 is derived from the MAC-16 address as required by 6LoWPAN header compression, collisions are likely to happen even in relatively small networks, i.e., around 300 nodes.

To avoid address collisions, the network coordinator has to avoid all values of the parameter r that will lead to such collisions. The *DODAG Version Number* can be used for this purpose, since it is not mandatory to use consecutive numbers when increasing the DODAG Versions. Using the *DODAG Version Number* has, however, some notable drawbacks, mainly due to the length of this field: only 8 bits. Moreover, the address must not fall in the reserved group of addresses group, and (at least) one bit have to be reserved to recognize the address before and after an address change. As a consequence, the number of available address space is reduced, leading to a higher collision probability.

The reserved address problem is easily solved by letting nodes with a reserved address perform a new AShA evaluation with an extended METAindex, e.g., by having an internal counter concatenated to the *METAindex* such as

$$r = \text{counter} \,||\, \text{METAindex}; \tag{7}$$

the counter is increased by 1 until a non-reserved address is found.

Nevertheless, the number of available *DODAG Version Numbers* is very limited, and with a large number of nodes in the network, a collision could be so frequent to leave only a few available *DODAG Version Numbers* to be used.

In order to solve this problem, we propose to use the DODAG Version Number as the Primary Index, due to the RPL internal mechanisms ensuring its network-wide spread, and a separate Secondary Index, to be carried either in the unused bits parts of the DIO messages (8 bits) or as a separate RPL Option Header. In the second case it is possible to use a larger Secondary Index. It should be noted that a typical DIO message is quite short (up to 56 bytes, when the 802.15.4 security is used). As a consequence, there is enough space in a single 802.15.4 message to carry the Secondary Index. As a matter of fact, we would suggest to use an Option Header for all but the simplest networks because 1) it does not require a non-standard implementation, and 2) it allows to choose the Secondary Index length dynamically.

Due to the fact that upward routes are mandatory, we use DIO messages to trigger AShA algorithm. Note that,

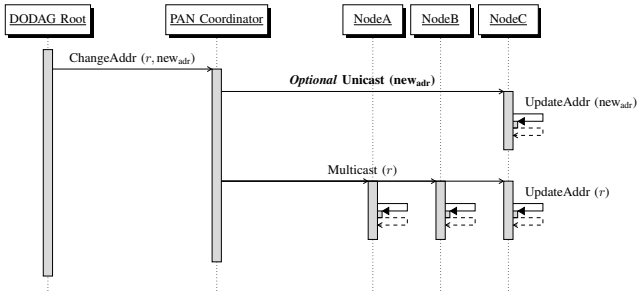

Fig. 5. Message sequence diagram for AShA address renew procedure.

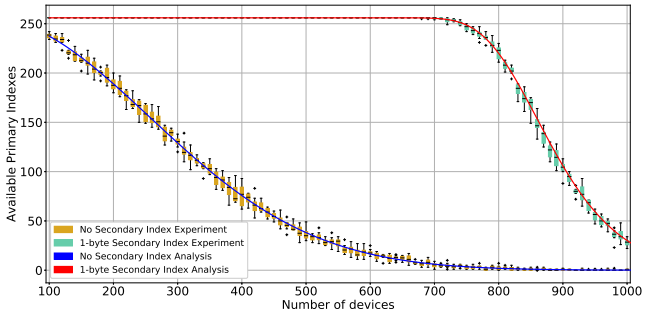

Fig. 6. Available DODAG version number with the Primary Index and Primary & Secondary Indexes.

in the extreme case of a shortage of viable alternatives, the Network Coordinator can always fallback to unicast messages to resolve collisions. Fig. 5 shows the worst-case sequence diagram of AShA algorithm: the DODAG Root evaluates the collisions, chooses the appropriate new values for *Secondary Index* and *DODAG Version Number*, and (eventually) sends a unicast address change message to some nodes. As shown in the figure, the DODAG Root also cooperates with the PAN Coordinator to update its association tables and allows a smooth address transition.

The network downtime during an address reconfiguration is null for a Secondary Index change, because the routing tables can be updated on-the-fly. Moreover, the old addresses can be still used, thanks to the bit used to indicate the 'old' and 'new' address set (before and after the shuffling). For a Primary Index change, the network downtime is equivalent to the one normally experienced in an RPL network for a DODAG Version change, and it is usually limited to a few seconds. However, the DODAG Version is periodically increased by RPL to ensure an optimal routing topology, and AShA does not add any further network downtime to this procedure. As a consequence, we can conclude that AShA does not add any downtime to the network.

VI. Simulation Results

To assess the validity of the proposed method, we prototyped an AShA server in Python. With a simulation campaign we evaluated the number of collisions in a DODAG Version Number sequence, i.e., how many DODAG Version Numbers are available depending on the number of nodes in the network. As outlined in the previous section, it is always possible to use unicast messages to resolve the collisions. However, avoiding unicast messages has a number of advantages: no overhead (address renew command is included in RPL DIO messages), less attack footprint (the attacker can not intercept the unicast message), the address change is faster, and no extra energy consumption is needed.

For the sake of simplicity, in the simulations we disabled the reserved addresses checks, and the full 16 bits range has been used.

Fig. 6 shows the comparison between using only the DODAG Version Number (8 bits) or also the Secondary Index (8 + 8 bits). Using only the DODAG Version Number, collisions happens already with 100 nodes, requiring either to avoid the use of some DODAG Version Numbers or to use unicast messages to resolve the collisions. On the contrary, the Secondary Index allows a broader range of possibilities, making it possible to use all the DODAG Version Numbers up to with 700 nodes.

TABLE I
DODAG Version Numbers Vs the number of nodes

	Number of nodes	
Available DODAGs	*r = 8*	*r = 16*
2/3 (171)	220	880
1/2 (128)	290	900
1/3 (85)	380	930

It is evident the perfect agreement between the experimental data and the one foreseen by eq. 5 and 6.

Table I reports the maximum allowed number of nodes in order to have at least one third, half, and two third of the DODAG Version Numbers available.

Fig. 7 shows the average number of collisions when only the Primary Index is used. As noted before, in case of collisions it is possible to use unicast messages to resolve collisions, but this solution is not energy efficient (it requires extra signaling), and requires extra reconfiguration time.

Fig. 8 reports the number of available Secondary Indexes if we just use 1-byte Secondary Index size. It is evident that the curve follows the same distribution shown in eq. (5). This is easily explained by considering that, for each Secondary Index, the Primary Index becomes a constant. As a consequence, they can be switched in the formulas.

Fig. 9 shows the effect of increasing the Secondary Index length. Although increasing the size of the Secondary Index

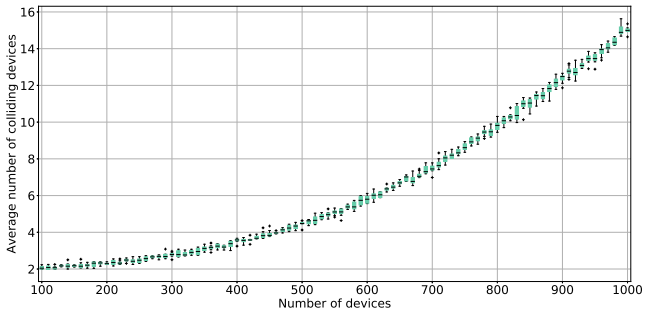

Fig. 7. Number of colliding nodes using only the Primary Index.

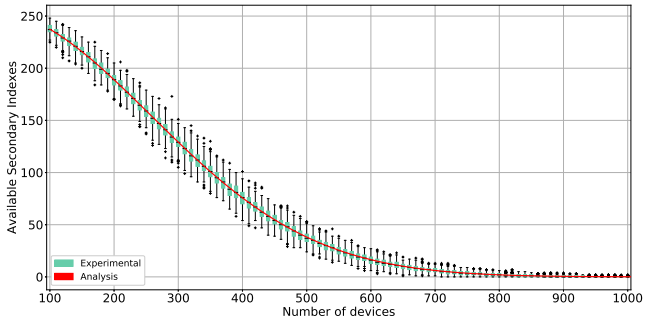

Fig. 8. Average number of available Secondary Indexes (1 bytes case).

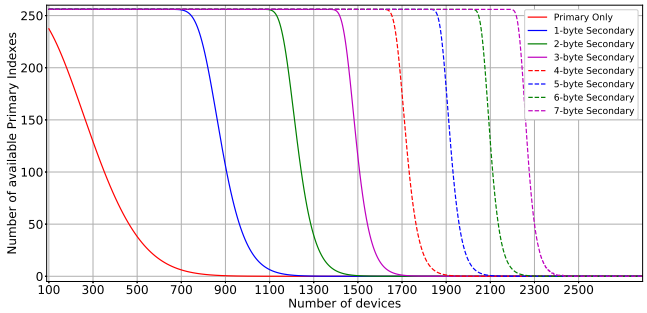

Fig. 9. Average number of available Primary Indexes varying the Secondary Index length.

allows a larger network size, the benefits becomes progressively smaller, and the computational costs for the Network Coordinator becomes larger.

Although it is possible to use any Secondary Index size, it is advisable to choose its correct size according to the actual number of nodes in the network, mainly to conserve energy and storage space.

Our suggestion is to choose a Secondary Index size that, according to the network size, guarantees an almost complete availability of Primary Indexes (see Fig. 9). Toward this end, it is possible to dynamically adjust the Secondary Index length, varying it according to the number of nodes in the network. In this way, it is possible to keep a minimal overhead while maintaining a high Primary Index availability.

VII. Conclusion

In this paper we presented Address Shuffling Algorithm with HMAC (AShA), a novel method allowing a fast, secure, and collision-free address renew in any (IPv6) network. We analyzed its features and limitations throughout an analytical formulation and with simulations, and we have shown how there is a match between our analysis and our simulation results. Our mathematical analysis can be used to predict AShA performances, thus serving as a guideline within the network security planning phase. With respect to the previous work, AShA has the benefits of eliminating any possible information leakage, such as the number of nodes in the network.

We argue that AShA is an excellent tool to provide the (network) confusion necessary in Moving Target Defense strategies, especially in Internet of Things scenarios, where energy efficiency is a strict requirement. AShA address renew messages can be piggybacked in normal routing maintenance (or any network management) semi-periodic message, reducing the network overhead to practically zero.

Acknowledgment

This work has been supported by the project “GAUChO - A Green Adaptive Fog Computing and Networking Architecture” funded by Ministero dell’Istruzione, dell’Università e della Ricerca (MIUR) - Progetti di Ricerca di Rilevante Interesse Nazionale (PRIN) Bando 2015 - grant 2015YPXH4W_004, by “ImpresaR&S 4.0” partially founded by POR FESR Toscana 2014-2020 asse 1 azione 1.1.5, and by NSF Award CNS-1647084.

References

[1] J. Manyika, M. Chui, P. Bisson, J. Woetzel, R. Dobbs, J. Bughin, and D. Aharon, “The Internet of Things: mapping the value beyond the hype,” McKinsey Global Institute, Tech. Rep., June 2015. [Online]. Available: https://www.mckinsey.com/business-functions/digital-mckinsey/our-insights/the-internet-of-things-the-value-of-digitizing-the-physical-world

[2] P. Datta and B. Sharma, “A survey on IoT architectures, protocols, security and smart city based applications,” in *2017 8th International Conference on Computing, Communication and Networking Technologies (ICCCNT)*, July 2017, pp. 1–5.

[3] D. Chemodanov, F. Esposito, A. Sukhov, P. Calyam, H. Trinh, and Z. Oraibi, “Agra: Ai-augmented geographic routing approach for IoT-based incident-supporting applications,” *Future Generation Computer Systems*, 2017. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0167739X17303965

[4] T. Borgohain, U. Kumar, and S. Sanyal, “Survey of security and privacy issues of Internet of Things,” *CoRR*, vol. abs/1501.02211, 2015. [Online]. Available: http://arxiv.org/abs/1501.02211

[5] G. Pulkkis, J. Karlsson, M. Westerlund, and J. Tana, “Secure and reliable Internet of Things systems for healthcare,” in *2017 IEEE 5th International Conference on Future Internet of Things and Cloud (FiCloud)*, Aug 2017, pp. 169–176.

[6] T. Pecorella, L. Pierucci, and F. Nizzi, ““network sentiment” framework to improve security and privacy for smart home,” *Future Internet*, vol. 10, no. 12, 2018. [Online]. Available: http://www.mdpi.com/1999-5903/10/12/125

[7] K. Sandlund, G. Pelletier, and L.-E. Jonsson, “The RObust Header Compression (ROHC) Framework,” RFC 5795 (Proposed Standard), RFC Editor, Fremont, CA, USA, pp. 1–41, Mar. 2010. [Online]. Available: https://www.rfc-editor.org/rfc/rfc5795.txt

[8] J. Hui and P. Thubert, “Compression Format for IPv6 Datagrams over IEEE 802.15.4-Based Networks,” RFC 6282 (Proposed Standard), RFC Editor, Fremont, CA, USA, pp. 1–24, Sep. 2011, updated by RFC 8066. [Online]. Available: https://www.rfc-editor.org/rfc/rfc6282.txt

[9] R. Housley and B. Aboba, “Guidance for Authentication, Authorization, and Accounting (AAA) Key Management,” RFC 4962 (Best Current Practice), RFC Editor, Fremont, CA, USA, pp. 1–23, Jul. 2007. [Online]. Available: https://www.rfc-editor.org/rfc/rfc4962.txt

[10] D. Thaler, “Privacy Considerations for IPv6 Adaptation-Layer Mechanisms,” RFC 8065 (Informational), RFC Editor, Fremont, CA, USA, pp. 1–10, Feb. 2017. [Online]. Available: https://www.rfc-editor.org/rfc/rfc8065.txt

[11] P. Wang, D. S. Reeves, and P. Ning, “Secure address auto-configuration for mobile ad hoc networks,” in *Mobile and Ubiquitous Systems: Networking and Services, 2005. MobiQuitous 2005. The Second Annual International Conference on*. IEEE, 2005, pp. 519–521.

[12] Centers for Medicare & Medicaid Services, “The Health Insurance Portability and Accountability Act of 1996 (HIPAA),” Online at http://www.cms.hhs.gov/hipaa/, 1996.

[13] J. Wang, J. Pan, F. Esposito, P. Calyam, Z. Yang, and P. Mohapatra, “Edge cloud offloading algorithms: Issues, methods, and perspectives,” *ACM Computing Surveys*, vol. pp, 2018. [Online]. Available: http://arxiv.org/abs/1806.06191

[14] Y. Hong, W. M. Liu, and L. Wang, "Privacy preserving smart meter streaming against information leakage of appliance status," *IEEE Transactions on Information Forensics and Security*, vol. 12, no. 9, pp. 2227–2241, Sept 2017.

[15] F. Ye and R. Pan, "A survey of addressing algorithms for wireless sensor networks," in *Wireless Communications, Networking and Mobile Computing, 2009. WiCom '09. 5th International Conference on*, Sept 2009, pp. 1–7.

[16] R. Droms, "Dynamic Host Configuration Protocol," RFC 2131 (Draft Standard), RFC Editor, Fremont, CA, USA, pp. 1–45, Mar. 1997, updated by RFCs 3396, 4361, 5494, 6842. [Online]. Available: https://www.rfc-editor.org/rfc/rfc2131.txt

[17] S. Thomson, T. Narten, and T. Jinmei, "IPv6 Stateless Address Autoconfiguration," RFC 4862 (Draft Standard), RFC Editor, Fremont, CA, USA, pp. 1–30, Sep. 2007, updated by RFC 7527. [Online]. Available: https://www.rfc-editor.org/rfc/rfc4862.txt

[18] E. Ancillotti, R. Bruno, M. Conti, and A. Pinizzotto, "Dynamic address autoconfiguration in hybrid ad hoc networks," *Pervasive and Mobile Computing*, vol. 5, no. 4, pp. 300–317, 2009.

[19] L. H. Yen and W. T. Tsai, "Flexible address configurations for tree-based ZigBee/IEEE 802.15.4 wireless networks," in *22nd International Conference on Advanced Information Networking and Applications (aina 2008)*, March 2008, pp. 395–402.

[20] S. Nesargi and R. Prakash, "MANETconf: configuration of hosts in a Mobile Ad Hoc Network," in *INFOCOM 2002. Twenty-First Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings. IEEE*, vol. 2, 2002, pp. 1059–1068 vol.2.

[21] C. E. Perkins, J. T. Malinen, R. Wakikawa, E. M. Belding-Royer, and Y. Sun, "Ad hoc address autoconfiguration," *draft-ietf-manet-autoconf-01.txt*, 2001.

[22] Y. Sun and E. M. Belding-Royer, "Dynamic address configuration in mobile ad hoc networks," University of California, Tech. Rep., 2003.

[23] D. Thaler, "Enabling Security/Privacy Addressing On 6LoWPAN Technologies," draft-thaler-6lo-privacy-addrs-00, Internet Engineering Task Force, Aug. 2015. [Online]. Available: https://tools.ietf.org/pdf/draft-thaler-6lo-privacy-addrs-00.pdf

[24] L. Brilli, T. Pecorella, L. Pierucci, and R. Fantacci, "A novel 6LoWPAN-ND extension to enhance privacy in IEEE 802.15.4 networks," in *Globecom 2016*. IEEE, 345 E 47TH ST, NEW YORK, NY 10017 USA, Dec. 2016, pp. 1–5.

[25] C. Lei, H. Zhang, J. Tan, Y. Zhang, and X. Liu, "Moving target defense techniques: A survey," *Security and Communication Networks*, vol. 2018, pp. 1–25, 2018. [Online]. Available: https://doi.org/10.1155/2018/3759626

[26] *MTD '18: Proceedings of the 2018 Workshop on Moving Target Defense*. New York, NY, USA: ACM, 2018. [Online]. Available: http://csis.gmu.edu/MTD-2018/

[27] Y. Sun and E. M. Belding-Royer, "A study of dynamic addressing techniques in mobile ad hoc networks," *Wireless Communications and Mobile Computing*, vol. 4, no. 3, pp. 315–329, 2004.

[28] H. Krawczyk, M. Bellare, and R. Canetti, "HMAC: Keyed-Hashing for Message Authentication," RFC 2104 (Informational), RFC Editor, Fremont, CA, USA, pp. 1–11, Feb. 1997, updated by RFC 6151. [Online]. Available: https://www.rfc-editor.org/rfc/rfc2104.txt

[29] N. Moore, "Optimistic Duplicate Address Detection (DAD) for IPv6," RFC 4429 (Proposed Standard), RFC Editor, Fremont, CA, USA, pp. 1–17, Apr. 2006, updated by RFC 7527. [Online]. Available: https://www.rfc-editor.org/rfc/rfc4429.txt

[30] "IEEE Standard for Low-Rate Wireless Networks," *IEEE Std 802.15.4-2015 (Revision of IEEE Std 802.15.4-2011)*, pp. 1–709, April 2016.

[31] G. Montenegro, N. Kushalnagar, J. Hui, and D. Culler, "Transmission of IPv6 Packets over IEEE 802.15.4 Networks," RFC 4944 (Proposed Standard), RFC Editor, Fremont, CA, USA, pp. 1–30, Sep. 2007, updated by RFCs 6282, 6775, 8025, 8066. [Online]. Available: https://www.rfc-editor.org/rfc/rfc4944.txt

[32] T. Pecorella, L. Brilli, and L. Mucchi, "The Role of Physical Layer Security in IoT: A Novel Perspective," *Information*, vol. 7, no. 3, p. 49, Aug. 2016. [Online]. Available: http://www.mdpi.com/2078-2489/7/3/49

[33] H. Krawczyk, "Cryptographic extraction and key derivation: The HKDF scheme," in *Advances in Cryptology – CRYPTO 2010*, T. Rabin, Ed. Berlin, Heidelberg: Springer Berlin Heidelberg, 2010, pp. 631–648.

[34] H. Krawczyk and P. Eronen, "HMAC-based Extract-and-Expand Key Derivation Function (HKDF)," RFC 5869 (Informational), RFC Editor, Fremont, CA, USA, pp. 1–14, May 2010. [Online]. Available: https://www.rfc-editor.org/rfc/rfc5869.txt

[35] N. Kushalnagar, G. Montenegro, and C. Schumacher, "IPv6 over Low-Power Wireless Personal Area Networks (6LoWPANs): Overview, Assumptions, Problem Statement, and Goals," RFC 4919 (Informational), RFC Editor, Fremont, CA, USA, pp. 1–12, Aug. 2007. [Online]. Available: https://www.rfc-editor.org/rfc/rfc4919.txt

[36] Z. Shelby, S. Chakrabarti, E. Nordmark, and C. Bormann, "Neighbor Discovery Optimization for IPv6 over Low-Power Wireless Personal Area Networks (6LoWPANs)," RFC 6775 (Proposed Standard), RFC Editor, Fremont, CA, USA, pp. 1–55, Nov. 2012. [Online]. Available: https://www.rfc-editor.org/rfc/rfc6775.txt

[37] T. Narten, E. Nordmark, W. Simpson, and H. Soliman, "Neighbor Discovery for IP version 6 (IPv6)," RFC 4861 (Draft Standard), RFC Editor, Fremont, CA, USA, pp. 1–97, Sep. 2007, updated by RFCs 5942, 6980, 7048, 7527, 7559, 8028. [Online]. Available: https://www.rfc-editor.org/rfc/rfc4861.txt

[38] T. Winter, P. Thubert, A. Brandt, J. Hui, R. Kelsey, P. Levis, K. Pister, R. Struik, J. Vasseur, and R. Alexander, "RPL: IPv6 Routing Protocol for Low-Power and Lossy Networks," RFC 6550 (Proposed Standard), RFC Editor, Fremont, CA, USA, pp. 1–157, Mar. 2012. [Online]. Available: https://www.rfc-editor.org/rfc/rfc6550.txt