Lightweight Address Hopping for Defending the IPv6 IoT

Judmayer Aljosha
SBA Research
ajudmayer@sba-research.org

Ullrich Johanna
SBA Research
jullrich@sba-research.org

Merzdovnik Georg
SBA Research
gmerzdovnik@sba-research.org

Voyiatzis Artemios G.
SBA Research
avoyiatzis@sba-research.org

Weippl Edgar
SBA Research
eweippl@sba-research.org

ABSTRACT

The rapid deployment of IoT systems on the public Internet is not without concerns for the security and privacy of consumers. Security in IoT systems is often poorly engineered and engineering for privacy does not seem to be a concern for vendors at all. The combination of poor security hygiene and access to valuable knowledge renders IoT systems a much-sought target for attacks.

IoT systems are not only Internet-accessible but also play the role of servers according to the established client-server communication model and are thus configured with static and/or easily predictable IPv6 addresses, rendering them an easy target for attacks.

We present 6HOP, a novel addressing scheme for IoT devices. Our proposal is lightweight in operation, requires minimal administration overhead, and defends against reconnaissance attacks, address based correlation as well as denial-of-service attacks. 6HOP therefore exploits the ample address space available in IPv6 networks and provides effective protection this way.

CCS CONCEPTS

•**Computer systems organization** →**Embedded systems;** *Redundancy;* •**Networks** →Network reliability;

KEYWORDS

Internet of Things, IoT, IPv6, security, network security, address hopping, denial of service, reconnaisance

ACM Reference format:
Judmayer Aljosha, Ullrich Johanna, Merzdovnik Georg, Voyiatzis Artemios G., and Weippl Edgar. 2017. Lightweight Address Hopping for Defending the IPv6 IoT. In *Proceedings of ARES '17, Reggio Calabria, Italy, August 29-September 01, 2017,* 10 pages.
DOI: 10.1145/3098954.3098975

1 INTRODUCTION

The Internet of Things (IoT) is the most recent evolutionary step of the Internet and brings billions of physical, formerly stand-alone devices online. IoT devices come in a great variety of size and functionality but show also great diversity with respect to their computing capabilities.

IoT systems aim to provide an increased level of comfort, safety, and efficiency: Smart watches can monitor the heart rate, allow sending short messages, and play music, all by means of a single small wrist device. Smart thermostats can learn habits and preferences of their owner, can control the heating system accordingly, and allow to turn on the air condition remotely while commuting to home or office. At the same time, more and more connected vehicles are driving along the roads – it is assumed that by the year 2020 there will be a quarter billion of such cars [6].

The rapid deployment of IoT systems on the public Internet is not without concerns for the security and privacy of the consumers. More often than not, IoT systems appear to be poorly engineered regarding their security, posing additional threats to the operation of the Internet. This was clearly demonstrated in the recent case of Mirai, the first IoT DDoS botnet[1]: CCTV cameras and DVR's around the world were compromised and formed a botnet eventually launching denial-of-service attacks against popular websites.

Security being an afterthought, privacy does not seem to be a concern for IoT system vendors at all. However, all these devices sense and interact with our surroundings. They continuously collect, process, and transmit detailed information about our lives to cloud infrastructures in remote locations, i. e., sensitive data unprotectedly travels the public Internet. The combination of poor security hygiene and access to valuable knowledge render IoT systems a first-class target for privacy attacks [25].

IPv6 and IoT go hand-in-hand today: for the IoT to realize its full potential, IPv6 provides enough addresses for the billions of devices to be connected and involved in human-to-machine and machine-to-machine (M2M) interactions. This is not feasible with IPv4: the address space is not big enough [13]. As of March 2017, IPv6 adoption surpassed the 16% mark among the Google users[2]. Despite the rising adoption trend, most of the available IPv6 addresses will still go unused for all current and envisioned IoT usage scenarios. Hence, the question arises: how can we utilize the address space offered by IPv6 to improve security and privacy in an IoT world? As an address has to be assigned in order to connect with the Internet anyway, secure and privacy-aware address configuration represents a lightweight mechanism for protection.

In this paper, we propose IPv6-Hopping (6HOP), a lightweight IPv6 addressing mechanism that is suitable for IoT scenarios. In

ARES '17, Reggio Calabria, Italy

978-1-4503-5257-4/17/08...$15.00
DOI: 10.1145/3098954.3098975

[1] https://www.incapsula.com/blog/malware-analysis-mirai-ddos-botnet.html
[2] https://www.google.com/intl/en/ipv6/statistics.html

6HOP, addresses are generated in a deterministic way by communication partners, but they appear random to third parties. This way, the partners are able to reconnect to each other whenever desired; but others are not able to pinpoint current endpoint information as addresses expire in a regular fashion.

Overall, our 6HOP proposal provides network-level mechanisms for:

- *Proactive security:* 6HOP does not disclose smart-home-specific IoT addresses for future attacks. Hence, any IPv6 scanning activity during a reconnaissance phase cannot be utilized to launch targeted attacks at a later stage.
- *Network-layer anonymity:* 6HOP does not disclose the network topology, the network size, or the identity of the individual IoT-connected devices.
- *Privacy:* 6HOP does not allow client device tracking inside the same network or across different networks.

The remainder of the paper is organized as follows: Section 2 outlines our IoT usage scenario and the associated threat model. Section 3 introduces the design of IPv6-Hopping (6HOP) and details the system variants and protocol operations. Section 4 and 5 provide the privacy and security evaluation for 6HOP. Section 6 discusses existing IPv6 addressing mechanisms with respect to their capability of protection as well as related approaches from the IPv4 world and compares them with 6HOP. Finally, Section 7 concludes the paper and lays down future directions of work.

2 MOTIVATION

In the established client-server communication model of the Internet, IoT systems are not only Internet-accessible but also act as *servers*. This is necessary for automated M2M interactions as well as remote control and configuration by their operators or owners. It is quite common that IoT systems are configured with a practically static IPv6 address to better serve their role. At the same time, a large class of IoT systems are resource-limited devices and cannot employ advanced network security protections. This combination makes IoT systems an easy target for attacks, i.e., *sitting ducks*.

Server addresses on the Internet are intended to be static, either manually assigned or following the *Modified EUI-Format*. Such addresses allow to find servers easily. The IoT challenges this assumption as IoT servers are intended to be used solely by a limited number of people, e.g., the residents of a smart home. Neither should these servers be found nor accessed by anybody else on the Internet.

IP-level addresses are valuable metadata that can be used by a malicious actor to launch attacks against these systems and the privacy of their users. Existing IPv6 addressing schemes do not cope sufficiently with these issues for servers, as they produce static or predictable address patterns. Even when randomization is employed, it relies on link-layer information (e.g., MAC addresses) that exhibit strong patterns [16].

2.1 Smart home usage scenario

We consider a smart home IoT usage and attack scenario as depicted in Figure 1. We assume two stakeholders, namely Alice and Eve. Alice lives in a smart home environment and operates multiple IoT devices in her residence, e.g., a smart fridge, some security cameras, a thermostat, and smart lightbulbs. For reasons of convenience, these devices allow remote control and maintenance over the Internet; Alice accesses the devices by means of her mobile device's Internet browser or a vendor-provided smartphone app.

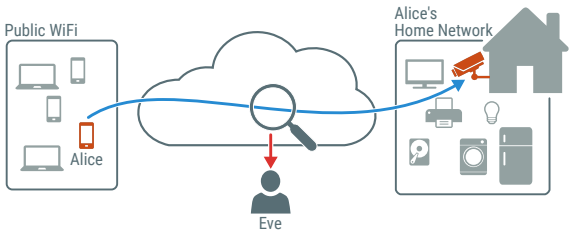

Figure 1: Threat model with a client on a public wireless network connecting back to a device at home

We assume full-IPv6 deployment in the public wireless network and Alice's home network. Thus, all smart devices have individual, globally-routable IPv6 addresses and are reachable from all over the Internet.

2.2 Attack scenarios

Eve has malicious intentions and wishes to violate the security of Alice and invade her privacy for nefarious purposes.

Eve aims to perform *active attacks* and strike the respective IoT device on the application layer, e.g., by exploiting poor authentication or other software vulnerabilities. Many IoT devices lack sophisticated security and are indeed highly vulnerable[3].

Launching an attack, however, requires prior reconnaissance, i.e., Eve must find the vulnerable devices in first place. As IoT devices are typically not announced in the Domain Name System (DNS), Eve rather has to actively scan addresses – a tedious but not entirely infeasible task with IPv6 [5, 7, 11, 21] – or, alternatively, extract addresses from captured network traffic. Hence, a first goal for our design is to increase the reconnaissance effort of the attacker. Ideally, by achieving this goal, we will prevent the attacker from reaching the (potentially vulnerable) application layer.

Eve also aims to perform *passive attacks* and inspect traffic in order to gain more information about Alice. Alice would connect to the same address whenever accessing a certain IoT device in her home as servers are intended to have a stable IPv6 address. In turn, Eve is able to attribute all these connection efforts to Alice by simply comparing target addresses. This line of action is commonly referred to as *address-based correlation*. It is especially outstanding among different types of traffic correlation as addresses are essential for packet delivery. They can neither be removed from packets nor encrypted.

Address-based correlation using *server* addresses is a consequence of the IoT device serving solely a limited number of clients in comparison to publicly accessible non-IoT servers. The IPv6 Privacy

[3] http://www.csoonline.com/article/3119765/security/hackers-found-47-new-vulnerabilities-in-23-iot-devices-at-def-con.html

Extension [19] considers it sufficient to scramble source (*client*) addresses. The rationale for this is that when connecting to non-IoT servers which serve a high number of different clients, the individual gets lost in the masses. Hence, a second goal for our design is to hinder address-based correlation by attackers exploiting server addresses. The attackers should not be able to identify communication endpoints based solely on the source and destination addresses of the latter.

We present in the next Section a dynamic IPv6 address shuffling scheme for the server side of the IoT that defends against the aforementioned attacks.

3 DESIGN OF 6HOP

6HOP specifies an algorithm deriving ephemeral addresses, ports and key information for both endpoints, i.e., Alice's smart phone and her domestic IoT server. Based on an initially exchanged secret all subsequent IPv6 addressing information is derived. The following scenario highlights the use of 6HOP.

In an initialization procedure, Alice's smart phone and the IoT server exchange a secret seed over the residential wireless network. As the wireless network is protected adequately, we consider this channel as secure and the exchanged secret as only known by the smart phone and the server.

The IoT server then calculates its address and transport layer port according to the deterministic algorithm specified by 6HOP, assigns itself the address, and listens for incoming requests at the transport layer port. 6HOP is a dynamic address format; the IoT server thus changes its address and port at a regular time interval, e.g., 24 hours, for purposes of protection, again following the 6HOP algorithm. Whenever Alice wishes to remotely access the IoT server, her smart phone follows the 6HOP algorithm including the initially exchanged secret as well into calculation. This way, it is able to infer the current destination address and port in order to reach the server.

Beyond, 6HOP includes further strategies for protection. First, the 6HOP algorithm allows not only to infer server addresses and ports, but also source addresses and ports. The smart phone has to use the latter in its connection attempts in order to authenticate itself as a legitimate client; requests from other source addresses and ports would be denied by the server upon receipt. The source address and port change in the same interval as the server address and port. Second, the 6HOP algorithm allows to infer secret keys to transmit authenticated and encrypted traffic which further increases the level of protection.

The remainder of this section describes the 6HOP algorithm for address derivation, describes how to move from one address to the next over time and introduces an overlapping address window for reasons of robustness.

3.1 Deriving address and key data

The core aspect of 6HOP is to derive addressing and authentication information from a shared ephemeral secret $e \in \mathcal{E}$ on every round of the protocol. In the first round of the protocol an initial secret seed s_0 is exchanged in a secure way. This initial seed is assumed to be chosen uniformly at random from $\mathcal{S}$ and is never used to derive addressing information directly. In the following paragraphs, we specify an algorithm that allows to infer IPv6 addresses and transport layer ports for both client and server as well as secret keys for encryption, thereby we model H as a random oracle.

s_n Bits	Length (Bits)	Param.
0-111	112	server IPv6 address suffix[4]
112-223	112	client IPv6 address suffix
224-239	16	server port
240-255	16	client port
256-383	128	key used for AE by A
384-511	128	key used for AE by B

Table 1: Partition of the 512-bit shared secret e_n into IPv6 address suffixes, ports, and keys used for authenticated encryption.

State s_n represents the intermediary value of a hash chain up to point n (see Equation 1).

$$\begin{aligned} s_1 &= H(s_0) \\ s_2 &= H\big(H(s_0)\big) \\ &\dots \\ s_n &= H^n(s_0) \end{aligned} \tag{1}$$

The ephemeral secret e_n can be calculated at both endpoints independently by applying a hash chain construction to s_0 or using the intermediate states s_n and s_{n-1} directly (see Equation 2).

$$\begin{aligned} e_1 &= H\Big(s_1 \oplus s_0\Big) \\ e_2 &= H\Big(s_2 \oplus s_1\Big) \\ &\dots \\ e_n &= H\Big(s_n \oplus s_{n-1}\Big) \end{aligned} \tag{2}$$

These subsequent shared values $e_1, e_2, \dots$ are used to derive *ephemeral IPv6 addresses, ports, and keys* for authenticated encryption (AE) on both endpoints of the 6HOP session.

Avoiding information leakage while keeping forward secrecy, the system distinguishes between ephemeral secrets $e_0, e_1, ..., e_n \in \mathcal{E}$ and states $s_0, s_1, ..., s_n \in \mathcal{S}$. The advantage of this construction is twofold. First, depending on the size of a sufficiently large *look-back* window w, all $\{e_0, \dots, e_{n-w}\}$ and $\{s_0, \dots, s_{n-(w+1)}\}$ values do not have to be maintained in storage. Since previous states are not necessary for computing upcoming ones, storing solely the last $s_{n-(w+1)}$ hash states provides forward secrecy for the system. Second, decoupling the hash chain state s from the actual shared ephemeral secret e that is used to derive addressing and authentication information, avoids that bits of the current state of the system, i.e., the current hash in the hash chain, are leaked to the outside over the used addresses and ports.

All shared ephemeral secrets $e \in \mathcal{E}$ are assumed to be of the same size and at least 512 bits. Since $|\mathcal{E}| = |\mathcal{S}|$ the size of each of these sets has to be at least 2^{512}. The derived shared ephemeral secret e is partitioned according to Table 1 in order to infer addresses, ports and encryption keys.

[4] IPv6 addresses are of 128 bits length; however, a node is usually not able to choose all bits independently. Network prefixes are typically determined by the Internet service

The method to initiate an update of the (hash chain) state, i. e., forwarding to the next address, needs to be synchronized among both endpoints. Otherwise, they would not be able to connect to each other. The following Section describes the synchronization requirements of this scheme in greater detail.

3.2 Forwarding to the next address

The idea of time-based hopping are addresses that change in a regular interval at both endpoints, i. e., the smart phone and the IoT server. For synchronization among the communication partners in agreeing to advance to the next IPv6 address the same point in time, we used a similar algorithm as defined for the time-based one-time password (TOTP) [18] as a baseline for our design. However, we do not include the time-based variable into the calculation of the ephemeral secret e_n, but only use it as a trigger mechanism to update the (hash chain) state s_n and eventually forward to the next address.

The update of the state is therefore triggered by a change of the time-based tick T. The tick T works as a counter and denotes the number of time steps of size X between the start time T_0 and the current time T_{now}. All these variables are measured in seconds. The included variables as well as the particular algorithm are described in the following paragraphs.

- *Current time T_{now}:* The current time as delivered by a real-time-clock.
- *Time step X:* The duration of one tick in seconds. The shorter the time step, the shorter ephemeral information like IPv6 addresses, ports and keys are valid. For example, the akin IPv6 Privacy Extension uses a life span of $86,400$ seconds (24 hours).
- *Time reference T_0:* The Unix time stamp of the initial connection. Having defined a time step X in the range of hours, it is not critical if the value of T_0 on the client differs from the value of T_0 on the server, since the renewal interval is a few hours anyway. Therefore it is not necessary to transmit T_0 explicitly. It can be deduced from the server when the initial connection is made.

The *time tick* T indicates when to update the local (hash chain) state. Therefore each implementation has to store at least the current value of T as well as T_0, X, and the current state s_n for every 6HOP session. From these values all future values can be derived. If a client for example wants to initiate a new TCP connection, it first computes T according to Equation 3.

$$T = \lfloor (\frac{T_{now} - T_0}{X}) \rfloor \tag{3}$$

Then the client checks the computed value T against the stored value T_{stored}. If $T_{stored} < T$, it sets $T_{stored} \leftarrow T$ and computes s_{n+1} and e_{n+1} to derive the appropriate addressing information to initiate the connection. If T_{stored} is equal to T, it uses the current addresses from s_n and e_n to initiate the connection.

provider (ISP) and announced by routers. Depending on the ISP, a domestic Internet connection receives a network prefix with a typical length between 48 and 64 bits; the latter are fixed, while the remaining bits can be chosen freely. In consequence, it remains sufficient to reserve 112 bits of the ephemeral secret per IPv6 address. A 6HOP node then takes the announced prefix, and fills the remaining bits with those from the ephemeral secret in order to form its 6HOP address.

3.3 Synchronization and overlapping window

Ideally, both endpoints hop to the next address, i.e., state, at the same point in time. However, due to network delays, clock discrepancies, and deviations in T_0, one of the following two cases might occur:

(1) The client's clock is too fast (i.e., the server clock is too slow) and the client tries to connect via an address that the server has not generated yet.
(2) The client's clock is too slow (i.e., the server clock is too fast). Then the client tries to connect to an already deprecated address.

To increase the resilience against such clock drifts, we propose a tolerance window of size w. This means that the ephemeral information, like IPv6 endpoint addresses, ports and keys, of the last w ticks will be kept active and reachable. This ensures connectivity and a smooth transition between consecutive IPv6 endpoint addresses. We assume that the clocks of both endpoints are synchronized to a degree that allows them to stay connected.

We suggest setting the overlapping window w and the time step X according to the security requirements of the respective 6HOP session but advise a *minimum* of $w = 1$. This means that there are always two valid IP addresses per 6HOP session, except for the very first interval. Minor clock drifts within the size of X should therefore not be an issue for 6HOP[5]. Thereby, a smooth transition for new TCP connections is ensured, while currently active TCP connections will not be reset but kept under the old IPv6 address until they close.

The time step X can also be configured in a way that the actual addresses appear like the IPv6 Privacy Extension, i.e., address generation in an interval of 24 hours. If the addresses should be changed every 24 hours, the tight synchronization of the clocks is less of an issue since an endpoint address would remain valid for 48 hours with $w = 1$.

Alternatively, X can also be reduced to increase the security. On the one hand, the shorter the time step, the less time an adversary has to strike a victim after identifying an endpoint address. On the other hand, a short time step requires tighter clock synchronization. To avoid problems originating from clock desynchronization, we propose a look-ahead window of minimum one time tick for 6HOP. This look-ahead window can also be used to reduce clock drift by readjusting on new connections within the next window.

4 SECURITY OF THE SHARED SECRET INFORMATION DERIVATION

A first attempt to break the security and privacy offered by 6HOP is to launch an attack against the secret information derivation function of 6HOP. If such an attack was successful, an attacker would know all future addresses, ports, and keys, effectively breaking the 6HOP mechanisms. We consider such attacks in the remainder of this section and infer an attacker's effort to do so. We also calculate the probabilities for address collisions, i.e., two nodes residing within the same subnetwork and assigning themselves the same addresses at the same point in time.

[5] The IPv6 privacy extension provides something similar: After its "preferred" lifetime of typically 24 hours, an address is kept for another "valid" lifetime in order to serve on-going transactions, but it is not used for new outgoing connections anymore.

4.1 Attack Design

Once knowing an internal hash state at an arbitrary point in time, an adversary is able to infer a victim's future 6HOP addresses, ports and keys. Indeed, these pieces of information are clips of the ephemeral secret e_n, as described in Table 1, which in turn are inferred from the hash state s_n and s_{n-1}. An adversary might be interested in gaining this hash state. The hash state might be leaked, e.g., by compromising one of the communication partners. Alternatively, a passive adversary is able to synchronize to the internal hash state by observing a number of successive identifiers, i.e., IPv6 addresses and/or ports[6], without actively compromising any of the communication partners.

Synchronization to the hash state is performed as follows. The adversary probes all possible values for s_{n-1}, and calculates s_n according to

$$s_n = H(s_{n-1}). \tag{4}$$

Then, she calculates the ephemeral secret e_n according to

$$e_n = H(s_n \oplus s_{n-1}), \tag{5}$$

and compares as many bits as possible. If the adversary is aware of both ports, and the interface identifiers, she is able to compare 160 bits. If she solely has the interface identifiers, she is able to compare only 128 bits.

The adversary excludes all inappropriate candidates; their secret keys do not correspond with the observed addresses and ports. The remainder are kept as suitable candidates, and their related hash states s_n are stored in a candidate set for the next iteration. The adversary reiterates with this reduced candidate set following the just described algorithm, and further reduces the candidate set by comparing with the addresses and ports that have been observed on the following day. She proceeds until the candidate set is reduced to a single hash state, this is the internal hash state of the algorithm. With this knowledge, the adversary is able to calculate all future addresses, ports, and keys.

In the remainder of this section, we discuss how many consecutive ephemeral addresses and ports have to be observed by an adversary in a row. In a second step, we quantify the effort for hashing.

4.2 Number of Observations

The adversary has to shrink the candidate set down to a single candidate. As she has to probe all possible values for s_0 having a length of 512 bits, the very first candidate set C_0 is of size 2^{512}. As the adversary is just able to compare the observed addresses and/or port numbers with the resulting ephemeral secrets e_0, more than a single candidate remains after the first iteration. Assuming a uniform distribution of hashes, the size of candidate set C_1 is

$$|C_1| = 2^{512} \cdot 2^{-b} \tag{6}$$

with b being the number of bits that the adversary is able to compare. The adversary is able to shrink the candidate set with every iteration. Its size in iteration n is calculated according to

[6]Observing successive identifier is a non-trivial task as an adversary cannot correlate identifiers by solely investigating these identifiers; but she might use some other form of metadata to do so. For example, in [23] the MAC address is used.

b	$T_{\min}$	
64	8	interface identifier of one communication partner
128	4	interface identifiers of both communication partners
80	7	interface identifier and port of one communication partner
160	4	interface identifier and port of both communication partners

Table 2: Minimal number of observations and probability of address collision in relation to observed attributes

$$|C_n| = 2^{512} \cdot 2^{-b \cdot n} \tag{7}$$

The adversary proceeds until a single candidate remains, i.e.,

$$|C_{T_{\min}}| = 2^{512} \cdot (2^{-b \cdot T_{\min}}) = 1 \tag{8}$$

In consequence, the number of consecutive addresses/ports that have to be observed by the adversary is

$$T_{\min} = \lceil \frac{512}{b} \rceil \tag{9}$$

Table 2 shows $T_{\min}$ for different scenarios. If the adversary is solely aware of the interface identifier of one communication partner, it takes eight observations. If she is aware of both interface identifiers, it reduces down to four observations; knowing interface identifiers and ports of both results in only four iterations.

4.3 Hashing Effort

Each iteration requires a substantial effort of hash computations. In particular, the adversary has to perform two hashes per candidate – one hash for calculating the next hash state s_n (cf. Equation 4), the other for calculating the ephemeral key e_n (cf. Equation 5). Thus, $|H_n|$, the number of hashes required for iteration n is

$$|H_n| = 2 \cdot |C_n| = 2 \cdot 2^{512} \cdot 2^{-b \cdot n} = 2^{513} \cdot 2^{-b \cdot n} \tag{10}$$

Summing up the number of hashes for all iterations leads to the following equation

$$|H| = \sum_{i=0}^{T_{\min}} |H_n| = 2^{513} \sum_{i=0}^{T_{\min}} 2^{-b \cdot i} \tag{11}$$

Assuming $2^{-l \cdot i} < 1$, the total amount of hashes is

$$|H| = 2^{513} \frac{2^{-b \cdot T_{\min}} - 1}{2^{-b} - 1} \tag{12}$$

and finally leads to the total time expenditure for brute forcing T_{Brute} assuming a hash rate r

$$T_{\text{Brute}} = \frac{2^{513}}{r} \frac{2^{-b \cdot T_{\min}} - 1}{2^{-l} - 1} \tag{13}$$

We measured $395 \cdot 10^6$ hashes per second on an AMD Radeon R9 290X GPU for SHA-512. Assuming the use of eight such GPUs in parallel leads to $3.2 \cdot 10^9$ hashes per second in total. In consequence, the described attack remains practically infeasible with today's commodity hardware.

Concluding, this attack does not appear economical from an adversary's point of view. For reconnaissance aims, it appears to be less effort to brute-force the IPv6 address space of 2^{128}. If the adversary aims to gain the encryption keys used for the authenticated encryption, it remains also less effort to brute-force the encryption directly.

4.4 Address Collisions

Address collisions refer to a situation where multiple nodes in a subnetwork assign the same IPv6 address. As multiple nodes listen to the same address, packet delivery becomes a mess; in consequence, address collisions have to be prevented in order to guarantee correct data transmission.

We consider two scenarios; the first considers the prevalence of two nodes A and B using 6HOP within a subnetwork, the second generalizes the previous result for an arbitrary number of nodes within a subnetwork. An address collision requires that addresses are equal in all of their bits; while the first bits of the network prefix are equal anyway when residing in the same subnetwork, the remainder x bits of the address are defined by 6HOP.

Scenario 1: Two nodes residing in a subnetwork use 6HOP. Assuming randomness of the hash function, the probability of both nodes assigning the same address at the same point in time calculates according to

$$P = 2^{-x} \tag{14}$$

with x describing the number of address bits that are calculated by means of 6HOP per node. Rearranging Equation 14 leads to the following expression

$$P = 1 - \frac{2^x \cdot (2^x - 1)}{2^{x \cdot 2}}. \tag{15}$$

Scenario 2: An arbitrary number of nodes reside in a subnetwork, and use 6HOP in order to generate their addresses. Assuming again randomness of the hash function, we are able to generalize Equation 15 to describe the probability of two nodes assigning the same address as follows

$$P = 1 - \frac{2^x \cdot (2^x - 1) \cdot \cdots \cdot (2^x - (N-1))}{2^{x \cdot N}} \tag{16}$$

with N being the total number of nodes within the respective subnetwork.

In conclusion, address collisions are highly improbable. However, in the rare case that a node observes such an address collision by using Duplicate Address Detection [17], we advise to automatically forward to the next address as defined by the 6HOP algorithm. The overlapping window parameter w, as defined in Section 3.3, successfully handles such a situation.

5 THREAT ANALYSIS

In this section, we discuss different kind of attacks and how the 6HOP mechanisms protects against these attacks. Thereby, we rely on the scenario defined in Section 2, and assume that the secret seed s_0 has been exchanged over a secure channel. In our analysis below, we model threats according to the STRIDE methodology [2, 20]. Therefore, we consider *spoofing, tampering, repudiation, information disclosure, denial-of-service* and *elevation of privilege* threats.

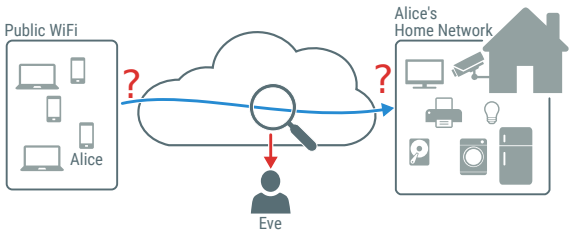

Figure 2: 6HOP prevents address-based correlation exploiting client addresses and correlation exploiting server addresses.

Further, we differentiate the attacks with regard to Eve's capabilities: (1) Eve is able to position herself *en-route*, i. e., on the traffic path between Alice's smart phone and her server as shown in Figure 2. Subsections 5.2 - 5.6 discuss attacks in this setting. (2) Eve is not able to place herself en-route; in consequence, she is only able to perform active probing as described in Subsection 5.1.

5.1 Active probing

Eve is not able to position herself en-route, i. e., on the traffic path between Alice and the server. Hence, she needs to perform active scanning in order to find potential victims like the IoT server. Scanning IPv6 addresses is a tedious but not entirely impossible task; currently known techniques of scanning mostly rely on implicit address patterns in order to reduce the search space [5, 7, 11, 21]. In consequence, it is rather unlikely that Eve finds a 6HOP address as they do not include any pattern, and rather appear random to third parties.

But even in the unlikely event that Eve finds a currently valid combination of an 6HOP address and transport layer port, her possibilities are limited. 6HOP requires the traffic to be authenticated and encrypted using one-time key pads. As Eve does not know the key, neither the ephemeral secret in order to infer the key, she is not able to initiate a new 6HOP session, or inject arbitrary data into the traffic flow. Moreover, the server is able to use source addresses as a form of authentication, and drop every packet not originating from a currently valid 6HOP source addresses. The latter further reduces Eve's probing capabilities.

5.2 Traffic inspection

If Eve is able to position herself within the traffic path between Alice's smart phone and the servers, she has attack capabilities that are more subtle than active probing. In a traffic inspection attack, Eve aims to read the exchanged data in order to gain sensitive information. Traffic inspection is encountered by 6HOP as the traffic is encrypted by the encryption keys that are derived from the current ephemeral key. Eve is not able to calculate the ephemeral key as she is lacking the secret seed that has been previously exchanged

over a secure channel, e. g., an adequately secured domestic wireless network.

5.3 Modify traffic

Eve may aim to perform a man-in-the-middle attack and modify the traffic between Alice's smart phone and the IoT server. However, 6HOP counters this attack as all traffic of an established 6HOP session is authenticated and encrypted using one-time keys that are inferred from the ephemeral secret. In consequence, Eve is unable to modify messages without being noticed and has no window of opportunity to launch man-in-the-middle attacks.

In case Eve wanted to exploit her power in order to deny communication between Alice's smart phone and the IoT server, she has to block all traffic involving the respective IPv6 network prefixes. The reason therefore lies in the changing endpoint addresses: Eve cannot selectively drop packets to a single endpoint, e. g., the IoT server, as the latter's address continuously changes and Eve is unable to identify successive addresses. Rather, she has to drop all packets going to or coming from these subnetworks. Denying any communication on the network level to a certain network prefix, e. g., by cutting the wire, cannot be prevent by 6HOP and is outside our threat model; however, it remains unlikely that such a coarse attacks remains unnoticed.

5.4 Replay traffic

Eve may record traffic of the 6HOP endpoints, and replay it at a latter point in time. As the addresses change over time, Eve is solely able to replay traffic in both direction as long as the included addresses, ports and keys remain valid. If 6HOP forwards to the next addresses, this capability of Eve is denied.

Nevertheless, she is able to reply traffic as long as the time-tick is not incremented. 6HOP does not protect higher application layer in all cases from the consequences of injection and replay attacks. The window of attack can be reduced by also decreasing the time-tick interval as well as the overlapping window size w, the application layer however remains responsible of detecting replay attacks, e.g., by the introduction of a counter.

5.5 Denial-of-service (DoS) attacks

Eve may aim to perform a denial-of-service attack against the IoT server by sending an increased amount of traffic. Our focus are denial-of-service attacks of increased traffic or targeted probing activity that can be handled by a consumer-grade home router (e.g., at the range of 100 Mbps) but will exhaust the resources of a home IoT device. We do not consider large scale distributed denial-of-service (DDoS) attacks producing several hundred Gbps of traffic as these kinds of attacks would paralyze domestic connections anyway.

In this scenario, we assume the current address of the IoT server has been recorded by Eve, e. g., by sniffing the traffic. Eve then launches a denial-of-service attack against this address. If a 6HOP endpoint is under a network-level denial-of-service attack, connectivity is automatically regained with the next time interval T_{n+1}, i. e., when hopping to the next address occurs. The struck address can effectively be blocked or redirected at upstream routers. Therefore, performing long-lasting, targeted probing or denial-of-service attacks is difficult as the adversary has to find successive addresses of a 6HOP session.

5.6 Traffic metadata analysis

Eve might aim to attribute Internet activities to Alice's smart phone and the IoT server by analyzing IPv6 addresses and transport laer ports. 6HOP protects against such address- and port-based traffic correlation attacks. Eve can deduce that two network prefixes are talking to each other, but she does not know for certain which devices are involved in the communication by solely looking at the used IPv6 addresses and ports.

The protection against address-based correlation offered by 6HOP is based on certain assumptions. If there is only a single device using 6HOP in an IPv6 subnet, all traffic coming from or going to this subnet can be attributed to this device i.e., the privacy set is too small to offer any protection. In other words, the protection against address-based correlation offered by 6HOP increases, the more clients/servers in the respective networks are using 6HOP or other formats of dynamic addressing. However, other dynamic addressing schemes, e.g., the IPv6 Privacy Extension suffer from the same limitation.

While address- and port-based correlation is mitigated by 6HOP, endpoints might still be fingerprinted by means of metadata analysis targeting network stack specific communication patterns. Different hosts might be identified, for example via their uptime deduced from TCP timestamps [14] or the frequency and time span over which different operating systems receive their updates. Such traffic artifacts can also be used by 6HOP peers to lay false traps and disguise themselves as other devices. The mitigation of such attacks and other techniques of traffic obfuscation are beyond the scope of this paper.

6 RELATED WORK

Shuffling addresses per se is not a novel idea for IPv6. Our research is based on two foundations: On the one hand, 6HOP is considered as a new address generation scheme for IPv6. Thus, we discuss existing approaches with respect to their quality of protection against denial-of-service and reconnaissance. On the other hand, alternatives for protection against these threats are available beyond IPv6; we also consider such approaches.

Many schemes for IPv6 address generation are available; with respect to stability over time, we distinguish three flavors of IPv6 addresses: (1) *static* addresses, (2) *semi-static* addresses, and (3) *dynamic* addresses. As all of them were designed with other goals in mind, it is not suprising that they are at most of limited benefit for protection against such attacks.

6.1 Static addresses

Static addresses remain effective over a long period of time, typically for a host's full lifetime. In today's IPv6 environment, two types of static addresses are prevalent. *Manually-assigned addresses* are mainly used at servers. Such addresses are individually configured by the operators. In contrast, addresses in *Modified EUI-Format* are inferred from the announced network prefix, the globally unique MAC address, and a fixed pattern [12].

Manually-assigned addresses frequently include patterns like port numbers of provided services or the respective IPv4 address [9], as a consequence of human intervention.

Modified EUI-Format addresses bear inherent patterns due to inclusion of the MAC address. These patterns allow to reduce the IPv6 address space for active probing [22]. They are rather an *enabler* for reconnaissance than a means of protection, and have became replaced as default identifiers recently [10].

Once an adversary gains a static address, she is able to perform a denial-of-service attack against the respective host. The attack traffic reaches the victim over the full attack period due to addresses stability. In consequence, static addresses can neither be considered to be a protection against reconnaissance nor against denial-of-service attacks.

6.2 Semi-static addresses

We consider addresses as *semi-static* in case the specification calls for changes in certain circumstances but these changes occur rarely in practice and are eventually of only limited benefit for protection against reconnaissance or denial-of-service attacks.

Cryptographically Generated Addresses [1] and *Semantically Opaque Identifiers* [8] include the network prefix into their deterministic algorithms for interface identifier generation. Both schemes include hash functions for calculation making the prevalence of address patterns unlikely. In consequence, they can be considered as protection against reconnaissance. However, they do not form an adequate means of protection against denial-of-service attacks. Addresses change *only* when moving to another network prefix. Hence, a viable protection would only be feasible through permanent (physical) movement.

6.3 Dynamic addresses

Dynamic addresses change at regular intervals. They are in principle the most promising for our scenario. First, they are typically generated in an automatic way and do not contain any patterns. Second, an addresses that is struck by a denial-of-service attack runs out automatically and a new one is generated. The adversary would have to find the respective host over and over again. As the addresses appear unrelated to each other, this remains a tedious task.

Known approaches of dynamic addresses are obviously the closest to our method. However, they are designed for different purposes. We discuss the cases of IPv6 Privacy Extension, MT6D, and non-IPv6 approaches in more detail in the next subsections.

6.4 IPv6 Privacy Extension

The IPv6 *Privacy Extension* creates addresses in a 24-hour interval to prevent traffic correlation of stationary and mobile clients [19]. Clients initiate outgoing connections but typically do not await incoming connections from others. At best, they choose the respective parts of the address in a random manner[7]. However, these unrelated addresses appear useless for our scenario. When applied to an IoT server, the latter would not be discovered by the legitimate clients either. 6HOP introduces a shared secret that allows the client to calculate the server's current address and reconnect.

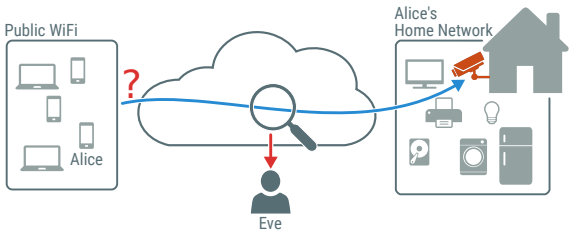

Figure 3: The Privacy Extension prevents address-based correlation exploiting client addresses, but not correlation exploiting server addresses.

In addition, the Privacy Extension is intended for clients; in consequence, transactions on the Internet cannot be attributed to the same client. Nevertheless, it remains possible to pinpoint all transactions to a certain server as it listens on a single static address. In our scenario, Eve would still be able to identify all connections attempts to the IoT devices by Alice by checking for the server address as a consequence of the limited amount of users accessing the IoT server. If the involved IoT devices use static addresses based on the Modified EUI-Format, it would even be possible to deduce what kind of device handled the respective connection based on the included MAC address [16].

6.5 MT6D

MT6D exploits permanently changing addresses as a means of moving target defense against denial-of-service attacks [4]. Such an approach would also protect against reconnaissance attacks. MT6D achieves this by *tunneling* IPv6 packets inside MT6D packets. This introduces an overhead of 62 bytes per IPv6 packet sent. The network performance (transfer speed and retransmissions) of MT6D is thus not satisfactory for modern applications [4].

MT6D is aimed towards connection-oriented upper layer protocols, like TCP. The sender (client) pre-calculates a receiver's (server) address every ten seconds based on an out-of-band negotiated secret in order to insert the then-valid address into the packet header. In this sense, *tight* time synchronization is crucial for the operation of MT6D.

Even though MT6D shuffles both server and client addresses, the client is able to infer the server address following a deterministic algorithm that includes a secret key. This addressing scheme bears however disadvantages. MT6D piggybacks the original communication into another layer of IPv6 in order to support connection-oriented protocols like TCP. This causes extra overhead and slow transfer rates. Also, the additional MT6D protocol headers foster traffic metadata analysis. Also, MT6D requires tight time synchronization which might be a barrier for resource constrained IoT devices.

Table 3 provides a summary of IPv6 address generation schemes. *Address Dynamics* refers to their lifetime; we distinguish static,

[7]Linux and Mac operating systems presumably do so; tough, the official algorithm appears flawed [23]. This fact however does not oppose the baseline of the *Privacy Extension*'s intention.

	Modified EUI-Format [12]	Semantically Opaque Addresses [8]	Cryto-graphically-Generated Addresses [1]	Privacy Extension [19]	MT6D [3]	6HOP
Address Dynamics	static	semi-static	semi-static	dynamic	dynamic	dynamic
Port Randomization	-	-	-	-	✓	✓
Reconnectivity	✓	✓	✓	-	✓	✓
Embeddedness	✓	✓	✓	✓	-	✓
Independence from Time Synchronization	✓	✓	✓	✓	-	✓

Table 3: Comparison of IPv6 addressing schemes

semi-static, and dynamic addresses as mentioned above. *Port Randomization* refers to the fact whether the address generation formats allows to randomize ports of the transport layer protocols, like UDP and TCP, in addition to the IPv6 addresses. *Reconnectivity* means that a host is able to reconnect to a certain host at a later point in time, i.e., the former can infer the address of the latter at this point in time. *Embeddedness* implies that the respective address generation schemes uses native IPv6, without any additional protocol headers. Finally, we highlight whether fine-grained time synchronization is needed.

6.6 Non-IPv6 approaches

An IPv4-based address shuffling defense for denial-of-service attacks is proposed in [15]. This approach requires multiple edge routers spanning multiple network segments of possibly different administrative domains. The sender decides the correct destination address according to a deterministic function. The respective router then checks for consistency and forwards the traffic to the apparent receiver.

In contrast to 6HOP, a rather small number of addresses is used due to the reliance on the available IPv4 address space. Furthermore, the proposal requires one edge router *per protected address*. Both these hinder the deployability in a consumer-grade IoT scenario. Smart homes will neither be assigned multiple public IPv4 addresses, given the IPv4 address scarcity, nor will they afford to install and maintain multiple edge routers.

Defenses that go beyond addressing are described in [24]. According to this classification, our proposal, 6HOP, is a destination-based approach of protection. Similar approaches are considered easy to deploy and comparably cheap. However, in the presence of an attack, the victim's resources are utilized (to a certain extent). In addition, the entity deploying the protection mechanism benefits from its deployment.

7 CONCLUSIONS AND FUTURE WORK

In this paper, we presented 6HOP, a lightweight address hopping scheme for defending IPv6-enabled IoT devices. Our proposal utilizes the ample IPv6 address space and defends against a wide range of attacks. It also integrates natively with the IPv6 protocol suite. Thus, 6HOP offers superior security characteristics and easier deployment, especially for smart homes and similar environments with a small (compared to open Internet) number of IoT servers.

Future directions of work include real-world studies on the effectiveness and the performance of 6HOP in IoT systems with different resource constraints regarding processing and storage capabilities. This includes studying scalability issues for serving a larger number of servers concurrently and alternative hopping techniques. Another interesting direction is to study the applicability of 6HOP in other usage scenarios like for example peer-to-peer systems.

ACKNOWLEDGMENTS

This work was supported partly by the Christian Doppler Forschungsgesellschaft (CDG) through Josef Ressel Center (JRC) projects TARGET and u'smile and the Austrian Research Promotion Agency (FFG) through projects SBA-K1, Bridge Early Stage A2Bit and CyPhySec.

REFERENCES

[1] T. Aura. 2005. Cryptographically Generated Addresses (CGA). RFC 3972 (Proposed Standard). (March 2005). http://www.ietf.org/rfc/rfc3972.txt Updated by RFCs 4581, 4982.
[2] Danny Dhillon. 2011. Developer-driven threat modeling: Lessons learned in the trenches. *IEEE Security & Privacy* 9, 4 (2011), 41–47.
[3] M. Dunlop, S. Groat, W. Urbanski, R. Marchany, and J. Tront. 2011. MT6D: A Moving Target IPv6 Defense. In *2011 - MILCOM 2011 Military Communications Conference*. 1321–1326. DOI:http://dx.doi.org/10.1109/MILCOM.2011.6127486
[4] Matthew Dunlop, Stephen Groat, William Urbanski, Randy Marchany, and Joseph Tront. 2012. The Blind Man's Bluff Approach to Security Using IPv6. *IEEE Security & Privacy* 10, 4 (2012), 35–43.
[5] Pawel Foremski, David Plonka, and Arthur Berger. 2016. Entropy/IP: Uncovering Structure in IPv6 Addresses. In *Proceedings of the 2016 ACM on Internet Measurement Conference (IMC '16)*. ACM, New York, NY, USA, 167–181.
[6] Gartner. 2015. A Quarter Billion Connected Vehicles Will Enable New Vehicle Services and Automated Driving Capabilities. http://www.gartner.com/newsroom/id/2970017. (January 2015). http://www.gartner.com/newsroom/id/2970017 Accessed: 2016-11-29.
[7] Oliver Gasser, Quirin Scheitle, Sebastian Gebhard, and Georg Carle. 2016. Scanning the IPv6 Internet: Towards a Comprehensive Hitlist. *CoRR* abs/1607.05179 (2016).
[8] F. Gont. 2014. A Method for Generating Semantically Opaque Interface Identifiers with IPv6 Stateless Address Autoconfiguration (SLAAC). RFC 7217 (Proposed Standard). (April 2014). http://www.ietf.org/rfc/rfc7217.txt
[9] F. Gont and T. Chown. 2015. *Network Reconnaissance in IPv6 Networks*. Internet-Draft. 1–35 pages. https://tools.ietf.org/html/draft-ietf-opsec-ipv6-host-scanning-08
[10] F. Gont, A. Cooper, D. Thaler, and W. Liu. 2017. Recommendation on Stable IPv6 Interface Identifiers. RFC 8064 (Proposed Standard). (Feb. 2017). http://www.ietf.org/rfc/rfc8064.txt
[11] Chris Grundemann. 2015. IPv6 Security Myth #4 – IPv6 Networks are Too Big to Scan. (2015). Available on http://www.internetsociety.org/deploy360/blog/2015/02/ipv6-security-myth-4-ipv6-networks-are-too-big-to-scan/.
[12] R. Hinden and S. Deering. 2006. IP Version 6 Addressing Architecture. RFC 4291 (Draft Standard). (Feb. 2006). http://www.ietf.org/rfc/rfc4291.txt Updated by RFCs 5952, 6052, 7136, 7346, 7371.
[13] Geoff Huston. 2016. IPv4 Address Report. (July 2016). http://www.potaroo.net/tools/ipv4/index.html

[14] V. Jacobson, R. Braden, and D. Borman. 1992. TCP Extensions for High Performance. RFC 1323 (Proposed Standard). (May 1992). http://www.ietf.org/rfc/rfc1323.txt Obsoleted by RFC 7323.

[15] Vladimir Krylov and Kirill Kravtsov. 2014. IP Fast Hopping Protocol Design. In *10th Central and Eastern European Software Engineering Conference in Russia (CEE-SECR '14)*. 11:1–11:5.

[16] Jeremy Martin, Erik Rye, and Robert Beverly. 2016. Decomposition of MAC Address Structure for Granular Device Inference. In *Proceedings of the 32nd Annual Computer Security Applications Conference (ACSAC 2016)*. ACM.

[17] N. Moore. 2006. Optimistic Duplicate Address Detection (DAD) for IPv6. RFC 4429 (Proposed Standard). (April 2006). http://www.ietf.org/rfc/rfc4429.txt Updated by RFC 7527.

[18] D. M'Raihi, S. Machani, M. Pei, and J. Rydell. 2011. TOTP: Time-Based One-Time Password Algorithm. RFC 6238 (Informational). (May 2011). http://www.ietf.org/rfc/rfc6238.txt

[19] T. Narten, R. Draves, and S. Krishnan. 2007. Privacy Extensions for Stateless Address Autoconfiguration in IPv6. RFC 4941 (Draft Standard). (Sept. 2007). http://www.ietf.org/rfc/rfc4941.txt

[20] Adam Shostack. 2008. Experiences threat modeling at Microsoft. In *Modeling Security Workshop. Dept. of Computing, Lancaster University, UK*.

[21] J. Ullrich, P. Kieseberg, K. Krombholz, and E. Weippl. 2015. On Reconnaissance with IPv6: A Pattern-Based Scanning Approach. In *2015 10th International Conference on Availability, Reliability and Security*. 186–192. DOI:http://dx.doi.org/10.1109/ARES.2015.48

[22] Johanna Ullrich, Katharina Krombholz, Heidelinde Hobel, Adrian Dabrowski, and Edgar Weippl. 2014. IPv6 Security: Attacks and Countermeasures in a Nutshell. In *USENIX Workshop on Offensive Technologies (WOOT)*. USENIX Association, San Diego, CA. https://www.usenix.org/conference/woot14/workshop-program/presentation/ullrich

[23] Johanna Ullrich and Edgar Weippl. 2015. Privacy is Not an Option: Attacking the IPv6 Privacy Extension. In *International Symposium on Research in Attacks, Intrusions, and Defenses (RAID)*. 448–468.

[24] S. T. Zargar, J. Joshi, and D. Tipper. 2013. A Survey of Defense Mechanisms Against Distributed Denial of Service (DDoS) Flooding Attacks. *IEEE Communications Surveys Tutorials* 15, 4 (Fourth 2013), 2046–2069.

[25] Jonathan L Zittrain, Matthew G Olsen, David O'Brien, and Bruce Schneier. 2016. Don't Panic: Making Progress on the "Going Dark" Debate. *Berkman Center Research Publication* 2016-1 (2016).