ALPHA: An Adaptive and Lightweight Protocol for Hop-by-hop Authentication

Tobias Heer, Stefan Götz,
Oscar Garcia Morchon, Klaus Wehrle
Distributed Systems Group
RWTH Aachen University, Germany
[heer, goetz, garcia, wehrle]@cs.rwth-aachen.de

ABSTRACT

Wireless multi-hop networks are particularly susceptible to attacks based on flooding and the interception, tampering with, and forging of packets. Thus, reliable communication in such networks quintessentially depends on mechanisms to verify the authenticity of network traffic and the identity of communicating peers. A major challenge to achieve this functionality are the tight resource constraints of such devices as smartphones, mesh- and sensor nodes with regard to CPU, memory, and energy. Since existing approaches suffer from significant drawbacks related to functionality and efficiency, we present in this paper ALPHA, an Adaptive and Lightweight Protocol for Hop-by-hop Authentication. ALPHA establishes a verifiable notion of identity for network traffic, based on computationally cheap hash functions, enabling end-to-end as well as hop-by-hop integrity protection for unicast traffic. Our evaluation shows that ALPHA is a generic security mechanism that makes full traffic authentication and secure middlebox signaling viable in resource-constrained multi-hop networks.

Keywords

Authentication, Integrity protection, Hash chains

1. INTRODUCTION

An ever-growing class of wireless devices constrained in their CPU power, available memory, and energy resources such as PDAs, smartphones, and pervasive wireless sensor nodes, enables the deployment of ubiquitous mobile and multi-hop networks. These devices are typically rolled out in untrusted or hostile environments with a wide range of attack vectors on packet communication. Thus, assuring its authenticity and integrity is of paramount importance on all network layers, e.g., for routing information, secure location updates of mobile hosts, code distribution, and for streaming sensed data or high-volume traffic.

In wireless multi-hop networks, a communication between end-hosts may involve a large number of forwarding nodes, making resource exhaustion attacks (e.g. targeting energy, bandwidth, and CPU resources) on any element of a communication path particularly effective. To limit the impact of this attack, it is vital to efficiently verify the authenticity of a message and its sender's identity to detect and drop forged or unauthorized messages early. Such a facility also allows on-path entities to authenticate data, e.g., for control and signaling data between end-hosts and forwarding nodes such as location updates from mobile devices. Together, forgery detection and data extraction form the basis for more complex services, such as rate and resource allocation within the network controlled by end-host but enforced by intermediate nodes.

Conventionally, light-weight end-to-end integrity protection and encryption are based on shared secrets and symmetric ciphers. However, these mechanisms cannot enable integrity checking on a hop-by-hop basis because forwarding nodes typically have no access to the shared secrets. Therefore, they cannot use these mechanisms to verify the authenticity of data and the identity of the communicating peers. Simply sharing the symmetric keys with forwarding nodes is not possible because malicious relays could use these keys to manipulate data in transit. Hence, packet manipulation and unauthorized transmission are only detected by the destination host and cannot be filtered by intermediate nodes. While public-key cryptography does not suffer from this limitation, it is computationally significantly more complex than the symmetric approaches. This overhead and the resulting impact on energy consumption and communication latency is prohibitive for per-packet verification in the vast majority of multi-hop scenarios.

Hash chains represent a practical basis for solving this problem as they are computationally efficient and are successfully employed in different specialized pro-

*This work was supported by the UMIC excellence cluster of RWTH Aachen University.

tocols, such as TESLA [18], CSA [4], ZCK [20], the Guy Fawkes Protocol [2], and WIMP [19]. However, existing solutions either lack on-path data verification or are too inefficient in wireless multi-hop networks for both infrequent low-volume and high-volume transfers. Moreover, they are designed for tightly restricted use-cases, making it difficult to apply them in a broader scope.

In this paper, we present and analyze ALPHA, an adaptive, flexible, and lightweight scheme for integrity protection and re-authentication based on hash chains. ALPHA provides end-to-end as well as hop-by-hop integrity protection for multi-hop wireless networks like mobile ad-hoc networks (MANETS), wireless mesh networks (WMNs), and wireless sensor networks (WSNs). Hence, it can replace traditional shared-secret-based end-to-end integrity protection, which cannot be authenticated by relays. We combine concepts from interactive signatures [19, 2, 4, 21] and Merkle Trees [15] that have been successfully applied in end-to-end-oriented network security. Our main contribution is to tie these techniques together into a single coherent and secure system that provides generic and efficient end-to-end *and* hop-by-hop integrity verification in unicast communication. We extend the set of existing hash-chain-based protocols by presenting three operational modes for ALPHA, allowing adaptation to the bandwidth, CPU, and memory capabilities of network nodes. Finally, we evaluate the efficiency and adaptiveness of the proposed scheme to illustrate its practical applicability in protecting three distinct scenarios and platforms: WSNs, WMNs, and lightweight mobile devices.

2. BACKGROUND AND RELATED WORK

Hash chains [10] suit well the design of efficient protocols as their computation is several orders of magnitude faster than public key cryptography. Since then, hash chains have been applied in a wide field of applications, such as the authentication and integrity protection of link-state routing updates [8, 5], integrity protection of multicast data [18], wormhole detection in mobile ad-hoc networks [9], and secure macro mobility and multihoming in IP networks [19]. In the remainder of this section, we review key concepts of hash-chain-based authentication and integrity protection and discuss related work in the field of hop-by-hop authentication.

2.1 Hash-chain-based Signatures

The fundamental idea behind hash chains is the iterated application of a cryptographic hash function H (e.g., SHA-1 or a block-cipher-based hash function) on a random *seed* value s. The first result $H(s) = h_1$ is used as input for the next round, yielding $H(H(s)) = H(h_1) = h_2$ until the hash chain has reached the desired length n. The last element of the chain h_n is called the *anchor*. The elements of this one-way hash chain are used in reverse order of creation, i.e., beginning with the anchor h_n and proceeding with h_{n-1}.

In terms of a protocol, the owner of a hash chain first exchanges the anchor with its peer. When required to authenticate itself, the owner reveals the next undisclosed hash chain element, and thus enables the receiver of the element to verify that it is in possession of the hash chain. Attaching a notion of identity to a hash chain, hosts can prove their identity for re-authentication by disclosing previously undisclosed elements of the chain as used by Hauser et al. [8]. This re-authentication property is important for mobile multi-hop networks as identities cannot be tied to non-cryptographic node characteristics, such as IP addresses, without security risks. Note that additional identity-providing techniques, such as public-key-authenticated hash chain anchors, are required for a strong assurance of identities.

There are three conceptually different approaches for signing and verifying messages with hash chains: one-time signatures, time-based, and interaction-based. In the remainder of this section, we give a short overview of time-based and interaction-based approaches and their properties. One-time signature schemes (e.g., [14, 23]), are not considered further because of their prohibitively high computational costs and large signature sizes.

2.1.1 *Time-based Signatures*

Cheung introduced *time-based signatures* for securing link-state routing [5]. Later, Perrig et al. proposed to use time-based hash-chain signatures in the Timed Efficient Stream Loss-tolerant Authentication (TESLA) [18] protocol. TESLA requires loose time synchronization to divide time into fixed-length time spans, where each epoch is linked to a different hash chain element. The ensuing data traffic is protected by a keyed-Hash Message Authentication Code (HMAC)[3], where the key is the element corresponding to the current epoch. μTESLA [11] was developed in order to improve the efficiency of TESLA for sensor networks by using only symmetric cryptography and restricting the number of authenticated senders in the network. This reduces the resource requirements for storing hash chains. However, time-based approaches provide only limited applicability to and adaptability for on-path authentication: First, jitter may lead to packets being delivered to a verifier after the corresponding hash-chain link was disclosed. The verifier consequently discards such packets. Thus, the minimum size of the time frame is determined by the maximum expected delay, drastically increasing the application-to-application latency in high variance networks, such as multi-hop wireless networks. Although Perrig et al. extended TESLA to adapt to different network latencies [17], this adaptation focuses on latency differences between multiple receivers of a multicast stream and does not apply to unicast communication. Second, time-based approaches reveal hash elements at a regular interval even when no payload is transferred. Thus, they incur computational overhead in networks with low or varying volume. Finally, existing time-based and end-to-end focused approaches do not take into account on-path integrity verification, and thus, are not applicable to this particular problem.

2.1.2 Interactive Signatures

Interactive hash chain-based signatures (IHC) exploit the interaction between a *signer* and a *verifier* to guarantee temporal separation between the generation of a signature, and the disclosure of the corresponding hash-chain element. Anderson et al. proposed the Guy Fawkes protocol [2] that uses interactive delayed secret disclosure for integrity protection and authentication of unicast streams. Bergadano et al. [4] proposed the Chained Stream Authentication (CSA) scheme, an interactive scheme for authenticating unicast and multicast streams. Torvinen and Ylitalo have shown that hash-chain-based signatures can be used for mobility and multihoming signaling in future IPv6 networks [19]. Weimerskirch and Westhoff [20] use an interactive approach in the Zero Common Knowledge protocol for re-recognition of communication partners. Yao et al. [21] use an interactive protocol to secure broadcast messages in sensor networks where a single source (the base station) sends identical messages to all nodes. Although the protocol achieves on-path authentication of messages, it does not provide efficient point-to-point communication between arbitrary nodes. IHC signatures in general do not require time synchronization and do not introduce a fixed delay until the receiver can verify the packets. Thus, they adapt well to scenarios with a widely varying network latency while their resource requirements are comparable to the requirements of TESLA.

Neither the time-based nor the interactive approaches lend themselves to securing point-to point communication in combination with on-path authentication of packets to suppress unsolicited traffic within the network. Moreover, the protocols lack adaptation capabilities regarding varying latency, bandwidth, and reliability requirements, and hence, each approach is restricted a specific use-case.

2.2 Hop-by-hop Authentication

LHAP [26, 12] and HEAP [1] were specifically designed for hop-by-hop authentication in MANETs. LHAP uses TESLA for bootstrapping trust relationships between nodes, and it uses authentication tokens when forwarding data packets. Lu and Pooch [12] propose HEAP, a system that builds on LHAP but uses a TESLA-like protocol for securing data transmission between two adjacent routers. HEAP uses pair-wise symmetric keys and a modified HMAC function to authenticate packets hop-by-hop. Gouda et al. present three protocols for *hop integrity protection* [6], in which symmetric keys between adjacent routers are used to identify injected and modified packets.

All of the aforementioned protocols aim at preventing outsider attacks by unauthorized senders. However, they cannot mitigate insider attacks such as forged or manipulated messages by otherwise trusted nodes. Protection against these attacks would require end-to-end integrity protection that can be verified on every hop.

Zhu et al. [25] and Ye et al. [22] solve the problem of efficient en-route verification with probabilistic approaches. However, both techniques are tightly coupled to a large sensor-network scenario with multiple cooperating sensors, sensing and sending the same information to a fixed sink (base station). Hence, the employed methods are not suitable for point-to-point communication between single hosts in networks of all sizes. Zhang et al. [24] use polynomial-based cryptography for authenticating packets in WSNs. Their approach assumes the presence of a central security server that provides keying-material to all nodes before deployment. Although this assumption is viable for many WSN scenarios, it is inapplicable to many dynamic and decentralized deployments.

3. DESIGN OF ALPHA

In this section, we present the design of the Adaptive and Lightweight Protocol for Hop-by-hop Authentication (ALPHA). ALPHA protects the communication between two arbitrary nodes in multi-hop networks. As depicted in Figure 1, it uses the notion of a protected path between these nodes. Before sending potentially large data packets, a small path reservation packet is sent to the destination, enabling the receiver and all intermediate nodes to efficiently check the integrity of the data packet. ALPHA is adaptive in the sense that it can be used for occasional signaling traffic as well as for high-volume data streams. Moreover it provides integrated support for reliable and unreliable data transmission. To this end, ALPHA provides two modes of operation (c.f. Section 3.2) and three transport mechanisms (c.f. Section 3.1 and Section 3.3) that can be combined to meet the requirements of different application scenarios and network capabilities.

3.1 Basic ALPHA Protocol

For a better understanding, we first give an overview of the basic ALPHA signature process before discussing extensions that enable the adaptation of ALPHA. The signature process takes place after an initial handshake in which the anchors of the hash chains are exchanged.

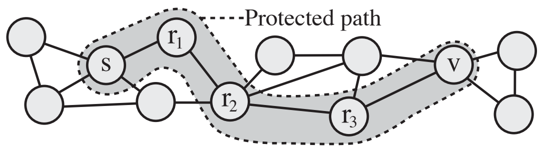


Figure 1: ALPHA dynamically establishes a protected path between a signer s and a verifier v over several relays r_i.

For the sake of clarity, we defer the handshake details to Section 3.4.

The goal of the signature process is to transmit an integrity-protected message m from a *signer* to a *verifier* in a way that lets *relays*[1] verify that m was (a) sent by a legitimate sender, (b) the sender is authorized to send m, and (c) m has not been altered by an attacker on the path.

The basic ALPHA signature scheme consists of a three-way packet exchange for each protected payload message m. Figure 2 depicts the packet exchange for the four-hop path in Figure 1. The ALPHA signature scheme belongs to the class of interactive hash chain signatures, hence, ALPHA uses deferred secret disclosure in combination with an interlocking scheme. The first packet announces a Message Authentication Code (MAC) M of m keyed with a fresh hash-chain element of the signer. In the second packet, the verifier acknowledges the receipt of the MAC and in the third packet, the signer sends m and discloses the hash chain element that was used as the MAC key. In the following we discuss the three-way signature process in detail.

Typically, an end-host acts both as a signer and a verifier on a bi-directional packet flow. Each host uses separate hash chains for signing outgoing and acknowledging incoming packets. Therefore, the shared security context between two hosts A and B consists of the respective anchors $\{h_n^{As}$, h_n^{Aa}, h_n^{Bs}, $h_n^{Ba}\}$. The two hash chains with superscript A are owned by host A while the other hash chains are owned by host B. Hosts use the first hash chain for signing data (i.e., it provides temporary keys for creating MACs) while they use the second chain for acknowledging the receipt of a message. Hence, the hash chains are denoted *signature chain* and *acknowledgment chain* and are signified by the second superscripts s and a. Each pair of a sender's signature chain and a receiver's acknowledgment chain protects a simplex channel. Hence, using the four-tuple protects a duplex-channel between the hosts. Note that a different set of hash chains is to be used for each path. In the remainder of the paper, we discuss the protection of such simplex channels between a signer S and a verifier V with the respective anchors being h_n^{Ss} and h_n^{Va} without loss of generality. By using two hash chains per host,

[1]Relays are forwarding nodes in a WMN, WSN, or MANET or other infrastructure elements, such as firewalls, that benefit from authenticating bypassing traffic.

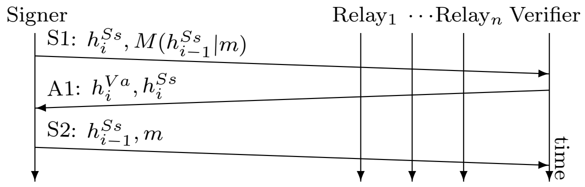

Signer
Relay$_1$ ···Relay$_n$ Verifier
S1: $h_i^{Ss}, M(h_{i-1}^{Ss}|m)$
A1: h_i^{Va}, h_i^{Ss}
S2: h_{i-1}^{Ss}, m
time

Figure 2: Basic ALPHA signature scheme. Relays can authenticate m before forwarding it.

ALPHA creates a full-duplex channel consisting of two simplex channels.

Each signature packet exchange is initiated with an S1 packet from the signer to the verifier. This packet fulfills three objectives. First, a fresh hash chain element of the signer's signature chain h_i^{Ss} identifies the signer. Second, a MAC keyed with the signer's next undisclosed signature chain element $M(h_{i-1}^{Ss}, m)$ ensures the integrity of m. Since attackers are not in possession of the undisclosed hash chain elements, they cannot forge valid MACs. The verifier and relays buffer the MAC until m and its key are disclosed through a subsequent S2 packet. Third, the S1 packet triggers the verifier to send an acknowledgment packet A1.

The A1 packet indicates that the verifier buffered the MAC and it expresses the willingness of the verifier to receive m. To authenticate the A1 packet, the verifier attaches the next undisclosed hash chain element of its acknowledgment chain h_i^{Va} to the A1. Similar to S1 packets, attackers cannot forge A1 packets as they are not in possession of h_i^{Va} before the verifier has received the S1 packet.

On receipt of a valid A1 packet, the signer discloses the key of the MAC h_{i-1}^{Ss} and the message m in the S2 data packet. With this key, the verifier and all relays that buffered $M(h_{i-1}^{Ss}, m)$ can check the integrity of m by recomputing the MAC. Tampering with the message m is ineffective because the verifier can check its validity against the tamper-proof MAC from the S1 packet.

3.1.1 *Efficient On-path Authentication*

For efficiency, similar to the broadcast authentication scheme in [21], ALPHA only transmits the MAC of a message in the first packet and sends m in the S2 packet, so the first packet only contains small hash values. The message m is still protected by the MAC and the temporal separation between the creation and delivery of the signature and the disclosure of the MAC key is still guaranteed. We refer to these signatures with delayed message disclosure as *pre-signatures*. Pre-signatures drastically reduce the amount of data buffered on verifiers and relays. Although the benefit for today's typical Internet end-hosts is marginal, this reduction makes hash-chain-based signatures feasible on low-end devices, such as sensor nodes. On forwarding devices in particular, pre-signatures offer significantly better scalability with the number of flows than regularly signed messages. Additionally, the lower buffer requirements render memory exhaustion attacks more difficult.

In the spirit of the Guy Fawkes protocol, pre-signatures in ALPHA do not reveal a message m until it can be verified. Thus, an attacker's window of opportunity to react to m and influence the verifier is reduced by a full round-trip time.

The relaying nodes on a path can verify the integrity and origin of a message if they have forwarded all previous signatures between the signer and the verifier. However, two colluding attackers[2] can replay forged signatures to a victim relay after diverting genuine signature packets around the victim (bypass attack). While the end-to-end integrity protection and the on-path filtering function of unsolicited packets are not affected by this attack (the second attacker must be located on the path behind the victim and it must express interest in receiving the replayed packets), the secure extraction of signed data by forwarding nodes suffers. The solution for preventing this attack is to keep the set of relaying nodes static throughout the use of a hash chain[3].

3.2 Acknowledgment Handling

To support a wide range of applications, ALPHA supports unreliable as well as reliable message transmission. In this section we present the two modes and discuss their properties.

3.2.1 Unreliable data transmission

For unreliable data transmission, ALPHA uses the basic three-way signature process. Due to the unreliable nature of the transmission, no explicit confirmation is provided when the verifier receives the S2 packet. Without additional security measures, a reformatting attack would be possible in which the hash chain element of an intercepted S2 packet and the following S1 packet could be used to generate a new S1 packet with a seemingly valid pre-signature. To prevent such an attack, ALPHA binds the elements of the signature chain to the purpose of either authenticating an S1 or providing the MAC key and authenticating the S2.

To this end, hash chains in ALPHA are constructed in the following way: $H_i = H(S1|H_{i-1})$ for odd occurrences of i and $H_i = H(S2|H_{i-1})$ for even occurrences of i. The values $S1$ and $S2$ are strings that make two sequential hash chain elements distinguishable. Hence, hash chain elements that are supposed to be used for MAC creation can be distinguished from hash chain ele-

[2]One situated on the path before, one behind a victim node.
[3]This can be achieved by additional local and lightweight security measures, such as interleaved message authentication codes or interleaved hash-chain-based authorization tokens between n-hop neighbors to detect and prevent the malicious bypassing of relays. The set of relay nodes can be fixed in the handshake and can be protected by either the receiver's hash chain signature or by public-key cryptography.

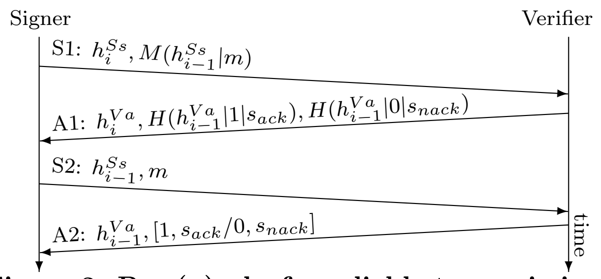

Signer
Verifier
S1: $h_i^{Ss}, M(h_{i-1}^{Ss}|m)$
A1: $h_i^{Va}, H(h_{i-1}^{Va}|1|s_{ack}), H(h_{i-1}^{Va}|0|s_{nack})$
S2: h_{i-1}^{Ss}, m
A2: $h_{i-1}^{Va}, [1, s_{ack}/0, s_{nack}]$
time

Figure 3: Pre-(n)acks for reliable transmissions

ments that are used for authenticating S1 packets. This enables a signer to send a new S1 packet immediately after receiving the A1 packet. A host that acts as signer and verifier can combine the packet transmissions of both directions and send A and S packets of independent simplex channels in the same packet.

3.2.2 Reliable data transmission

Each signed message from a signer to a verifier requires a three-way interactive signature process. Thus, using the same technique to send signed acknowledgements to allow for reliable data transmission would result in a total of six packets. We reduce this overhead to four packets by introducing *pre-acknowledgments* (*pre-acks*) and *pre-negative acknowledgments* (*pre-nacks*). Both are provided in the A1 packet and made verifiable via information in an additional A2 packet.

Figure 3 depicts the signature process with pre-acknowledgments. The verifier generates both the pre-ack and pre-nack value after it receives the pre-signature in the S1 packet. The pre-(n)ack values are hashes over the concatenation of three pieces of information. The first piece is the next undisclosed hash chain element of the verifier's acknowledgment chain h_{i-1}^{Va}. It is included to prevent attackers from forging the acknowledgments. The second piece is a fixed string for the pre-ack and a different fixed string for the pre-nack to make both values distinguishable (e.g., 0 and 1). The third piece of the pre-acknowledgment consists of two distinct secret random numbers s_{ack} and s_{nack}. These secrets prevent an attacker from computing a pre-nack from a pre-ack or vice versa after the corresponding hash chain element h_{i-1}^{Va} is disclosed.

If the signature of the message in S2 is valid, the verifier discloses the contents of the pre-ack, otherwise it discloses the contents of the pre-nack. It transmits the disclosed data in an A2 packet to the signer. The signer can verify that the packet was sent by the verifier by comparing the hashed result of h_{i-1}^{Va} to the hash chain element previously disclosed by the verifier. The string in the A2 packet (1 or 0 in the example) indicates an acknowledgment or negative acknowledgment for the data received by the verifier in the S2. The signer and relays can verify the validity of the ack or nack by hashing the concatenation of the hash chain element, the string, and the disclosed secret s_{ack} or s_{nack} and comparing it to the corresponding pre-(n)ack in the A1 packet.

To detect forgery attacks, temporal separation between the pre-(n)ack creation and the disclosure of the corresponding hash chain element is achieved by letting the signer discard pre-(n)acks in further A1 packets after it sent an S2 message. Also, using fresh random or pseudo-random secrets for every pre-(n)ack on the verifier thwarts replay attacks.

Depending on the transmitted payload, relays may also need to verify that a message was successfully received by the destination host. Examples for such payload are signaling protocols that require the relays to change their internal state (e.g. mobility management protocols for macro mobility and secure QoS signaling). To verify (n)acks on relays, the corresponding pre-(n)acks need to be buffered first.

Using pre-acks and pre-nacks offers two advantages in scenarios that require reliable data delivery. Firstly, they reduce the communication overhead in terms of required transmissions. Secondly, they reduce the latency for receiving the acknowledgement from three to two RTTs.

3.3 Bandwidth Adaptation

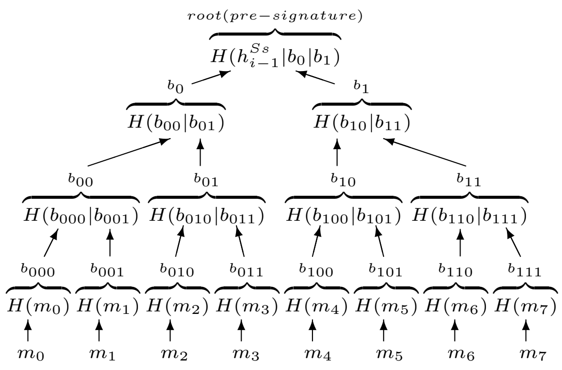
In its basic form, ALPHA signatures offer only limited support for transmitting large amounts of data because the strictly sequential packet exchange limits throughput. In this section, we present two modes for ALPHA that allow better adaptation to the bandwidth, memory, and computational resources of a multi-hop network. In particular, we evaluate the use of these modes for end-to-end and hop-by-hop authentication.

Taking relays into account, a solution for sending higher volumes of data must not substantially increase the memory requirements on relays. Moreover, due to the higher packet loss and the possibility of out-of order delivery in wireless networks, efficient verification of individual data packets must be ensured even if other data packets are lost or arrive out of sequence.

3.3.1 ALPHA-C: Cumulative transmission

Sending only a small pre-signature instead of the actual message in the S1 packet offers the possibility to send multiple pre-signatures in parallel. For high-volume data transfers, pre-signatures of multiple messages based on the same undisclosed hash chain element are transmitted to the verifier in a single S1 packet. The verifier then acknowledges the receipt of cumulative pre-signatures just as it would acknowledge an S1 with a single pre-signature. Thus, the signer can send the S2 packets for all pre-signed messages in parallel or in short succession without waiting for individual acknowledgments. We refer to this mode as ALPHA-C, standing for ALPHA with cumulative transmissions.

Although the sequence of S1 and A1 packets is strictly



$root(pre-signature)$
$H(h^{Ss}_{i-1}|b_0|b_1)$
b_0
b_1
$H(b_{00}|b_{01})$
$H(b_{10}|b_{11})$
b_{00}
b_{01}
b_{10}
b_{11}
$H(b_{000}|b_{001})$
$H(b_{010}|b_{011})$
$H(b_{100}|b_{101})$
$H(b_{110}|b_{111})$
b_{000} b_{001} b_{010} b_{011} b_{100} b_{101} b_{110} b_{111}
$H(m_0)$ $H(m_1)$ $H(m_2)$ $H(m_3)$ $H(m_4)$ $H(m_5)$ $H(m_6)$ $H(m_7)$
m_0 m_1 m_2 m_3 m_4 m_5 m_6 m_7

Figure 4: ALPHA-M: MT with 8 leaves.

sequential, the order of delivery of the S2 packets is not of importance. As the hash chain element h^{Ss}_{i-1} is present in all S2 packets, they can be verified by the verifier and relays independently. Packet loss of an S2 packet can either be handled by retransmitting the S2 packet or be ignored if retransmission is not required, e.g. for time-critical data like multimedia streams.

When using ALPHA-C, the verifier and relays buffer multiple pre-signatures. This requires additional buffer space of the size of up to the maximum packet payload per secured connection. The number of S2 packets increases linearly with the available buffer space of the relays and the verifier while the computational overhead for verifying an S2 packet is constant.

3.3.2 ALPHA-M: Pre-signed Merkle Tree

ALPHA-C enables higher throughput at the cost of increased buffer requirements but with constant computational overhead for verification. In the following, we discuss a complementary approach for sending n individually verifiable S2 packets with constant buffer size and computational cost increasing with $log_2(n)$.

ALPHA-M leverages trees of hashes, so called Merkle Trees (MTs) [15], to generate pre-signatures and pre-acks. An MT is a binary tree with the j^{th} leaf b_j containing the hash of the pre-image m_j and each node containing the hash of the concatenation of its two children. The root r of the MT is dependent on the contents of all leaves and, therefore, of all nodes in the tree. Thus, the modification of a single leaf or node results in a different root.

One application of MTs is to authenticate data: a signer constructs an MT by splitting the data into blocks m_j and using these as pre-images for the leaves b_j of the tree[4]. To authenticate a block m_j independently of other blocks, a verifier requires r, m_j and the set $\{B_c\}$ of complementary branches, which, for each level of the tree, consists of the sibling node of the nodes on the path from b_j to r. The verifier calculates the tree root r^* by reconstructing the path from $b_j = H(m_j)$ to

[4]Note that the leaf index j of the MT is given in a binary representation to emphasize the tree structure.

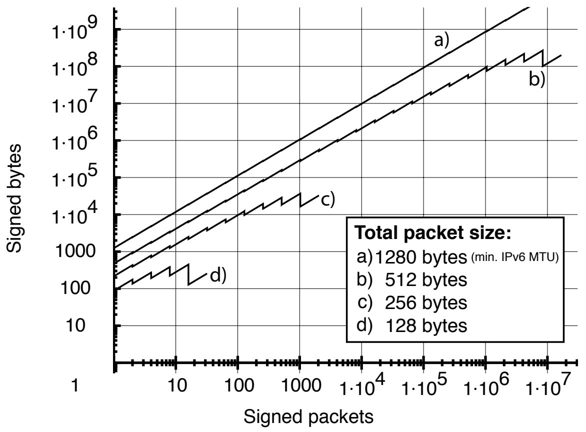


Figure 5: Signed bytes per S1 (20 B hash).

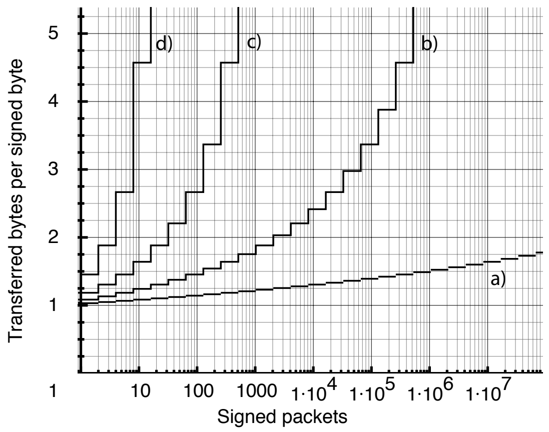


Figure 6: Signature overhead with varying MT size (20 B hash). See Fig. 5 for legend.

r from the nodes in $\{B_c\}$. A block m_j is authentic only if the known root r matches the computed value r^*.

As depicted in Figure 4, ALPHA-M applies MTs to the ALPHA scheme as follows: A signer constructs an MT with its buffered messages m_j serving as pre-images of the tree leaves. The root r of the MT serves as pre-signature in the S1 packet. The signer includes in the S1 packet the root $r = H(h^{Ss}_{i-1}|b_0|b_1)$ and a fresh element of its signature hash chain as authenticity token. The verifier authenticates S1, buffers r, and acknowledges the reception with the next element from its acknowledgment chain. Each message m_j is transmitted in an S2 packet along with the set $\{B_c\}$, such that m_j can be authenticated independently of other messages. With this design, the verifiability of individual messages by end hosts and on-path entities is preserved. Each S2 data packet can be verified individually. Compared to ALPHA-C, ALPHA-M requires significantly less pre-signature data to be buffered by relays and the verifier. However, the set $\{B_c\}$ that needs to be transmitted in each S2 increases by $log_2(n)$ with n as the number of data chunks $n = |m_j|$.

Assuming a fixed amount of pre-signature data in an S1 packet, the ALPHA-M approach provides a trade-off between (a) the amount of verifiable payload which can be transmitted in S2 packets en bloc per pre-signature in an S1/A1 exchange, (b) additional signature data transmitted along in the S2 payload packets, and (c) the computational complexity of S2 verification by relays or the verifier. To quantify this trade-off, we calculate s_{total}, the amount of payload that can be transmitted with a *single* pre-signature. We assume fixed-sized packets providing s_{packet} bytes of payload space to the ALPHA mechanism, and MT nodes (i.e., a hash output) of size s_h bytes. With n as the total number of S2 packets, the following holds:

$$s_{total} = n \cdot (s_{packet} - s_h(\lceil log_2(n) \rceil + 1)) \qquad (1)$$

The result for different packet sizes and tree sizes is depicted in Figure 5. With an increasing number of data chunks (S2 packets) per pre-signature, $\{B_c\}$ grows logarithmically, and thus, the signature size increases. Hence, when the $|\{B_c\}| \cdot s_h$ approaches the maximum packet size s_{packet}, the transferable payload drops with every new level of the MT, leading to the see-saw patterns in Figure 5.

Figure 6 depicts the ratio of sent bytes per payload byte which is of particular interest on energy-constrained devices. It illustrates that the overhead of signature data is lower for larger packets. Thus, the amount of total payload data covered by an S1 pre-signature is practically limited by the ratio of payload to signature data in each S2 packet, which depends strongly on the size of S2 packets.

In comparison with ALPHA and ALPHA-C, ALPHA-M introduces additional hash computations. Constructing the MT on the signer adds an overhead of $n-1$ hash calculations for generating the tree nodes to hashing the n messages as in the original pre-signature scheme. Using ALPHA-M requires the verifier and relays to compute $\lceil log_2(n) \rceil - 1$ additional hash computations for reconstructing the path to the root node of the MT. Note that all these additional hash computations are performed on the fixed-length concatenation of two hash outputs. Section 4 gives a detailed comparison of the computational cost of the three modes.

Covering a set $\{M\}$ of $n = |\{M\}|$ packets $m_j \in \{M\}$ with a single pre-signature establishes a temporal dependency between the packets. Thus, the signer needs to buffer a complete set of n packets before creating the pre-signed root of the MT. Hence, the transmission of any given m_j is delayed until all $m_j \in \{M\}$ are available at the signer, the MT is constructed, and the S1/A1 exchange has completed. Consequently, the stronger the latency requirements of a transmission, the smaller n needs to be chosen. However, the vast majority of communication in sensor-, mesh-, and ad-hoc networks does not have such strict requirements and can thus benefit from the efficiency gains of ALPHA-M. In fact, ALPHA-M and ALPHA-C signatures are well suited for wireless multi-hop networks because varying

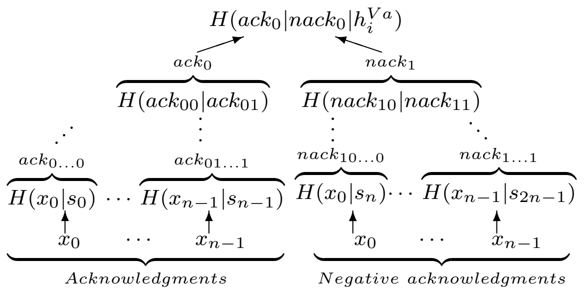


Figure 7: Merkle Tree for acknowledgments.

latency, out-of-order delivery, and high loss rates can be tolerated due to the individual verifiability of each S2 packet. ALPHA-C and ALPHA-M can be used in combination. Delivering multiple MT roots per S1 packet makes possible a reduction of the computational cost for verifying $\{B_c\}$ or enables the sender to send a larger number of S2 packets with constant cost. However, it requires larger buffering capabilities from relays.

3.3.3 Reliable data transmission with ALPHA-M

Using Merkle Trees obsoletes the pre-acknowledgment optimization presented in Section 3.2 because the required number of pre-(n)acks per S1 grows exponentially with an increasing MT depth. To still be able to selectively acknowledge every m_j, ALPHA-M uses an MT construct for generating pre-acks. As illustrated in Figure 7, pre-acks and pre-nacks are placed in the leaves of an Acknowledgment Merkle Tree (AMT), requiring a tree with $2n$ leaves for acknowledging n messages. Each leaf of the AMT contains a secret s_i and an index x_i, which identifies the packet m_j. The secret must be distinct for each leaf of the tree. Values from the *left* AMT branch are interpreted as acks and those from the *right* branch as nacks. The undisclosed hash chain element h_{i-1}^{Va} authenticates the root and prevents forgery. A2 messages contain the index x_i, the secret s_i, and the set of MT nodes $\{B_c\}$ necessary to compute the root from the leaf. The signer and each relay can verify each acknowledgment individually. Moreover, an AMT can enable retransmission schemes as selective repeat and go-back-n for ALPHA-M.

3.4 ALPHA Bootstrapping

Bootstrapping is the process of making hash chain anchors known to verifiers and relays on a path. Due to the wide applicability of ALPHA, we do not define a specific bootstrapping process but discuss the options of static vs. dynamic bootstrapping and unprotected vs. protected bootstrapping.

In the beginning of a communication association, the signer and the verifier need to exchange their hash chain elements for the *signature* and *acknowledgment chains.* This process can either be performed before deploying a network (e.g. WSNs) or when the network is already operational (e.g. for MANETs and WMNs). For preconfigured scenarios, such as static wireless sensor networks, base stations can provide nodes with pair-wise anchors. For dynamic scenarios, senders directly exchange their anchors in a handshake procedure on demand. If required, additional security measures against bypassing of relays (cf. Section 3.1.1) should also be bootstrapped during the handshake or when sending the first S1 packet.

An *unprotected handshake* provides each peer of a security association with an ephemeral anonymous identity that is only of use in the corresponding association. Even with an anonymous identity, hosts can use ALPHA to securely signal changes concerning an association (e.g. signaling new IP addresses, throttling the transmission rate, closing an association, etc.) to their peers. Relays learn the hash chain elements or anchors by observing a handshake.

A *protected handshake* binds hash chains to strong cryptographic identities (e.g., public-key-based certificates) and vice versa, which allows for identifying hosts (e.g., insiders and outsiders) or certain roles (e.g., coordinator, server, client). To protect bootstrapping, the anchor of a hash chain is signed with signatures based on asymmetric cryptography, such as RSA, DSA, and Elliptic Curve Cryptography (ECC). Because of the high resource requirements of asymmetric cryptography, ALPHA explicitly limits its use to this bootstrapping process. For strong hop-by-hop authentication towards relays, the public-key signature of the sender needs to be verified by each relay for bootstrapping and re-validated each time the set of relays changes. Due to the CPU complexity and energy consumption imposed by such cryptographic operations, such a strong hop-by-hop authentication can be assumed to be prohibitively resource intensive for MANETs with their frequently changing routes. However, it may be feasible for WSNs and WMNs in which routes fluctuate only occasionally.

3.5 ALPHA Strengths and Limitations

ALPHA provides adaptive end-to-end as well as hop-by hop integrity protection. Hence, it can replace traditional shared-secret-based end-to-end integrity protection mechanisms, which relays cannot verify. Moreover, relays can filter forged data and securely extract authentic information from the S2 packets, enabling them to react to the content of protected control messages. Therefore, ALPHA can also be used as a building block for secure signaling between end-hosts and relays.

ALPHA helps to mitigate flooding attacks since receivers can explicitly state whether or not they are willing to receive data from a sender by providing or denying an A1 packet. When the first relay on the communication path enforces this decision, unsolicited data

Table 1: Hash computations for processing one message. ALPHA-C and -M send n messages per S1.

	ALPHA			ALPHA-C			ALPHA-M		
	Signer	Verifier	Relay	Signer	Verifier	Relay	Signer	Verifier	Relay
Signature	1^*	1^*	1^*	1^*	1^*	1^*	$1^* + 2 - \frac{1}{n}$	$1^* + log_2(n)$	$1^* + log_2(n)$
HC create	2^+	2^+	0	$\frac{2}{n}^+$	$\frac{2}{n}^+$	0	$\frac{2}{n}^+$	$\frac{2}{n}^+$	0
HC verify	1	1	1	$\frac{1}{n}$	$\frac{1}{n}$	$\frac{1}{n}$	$\frac{1}{n}$	$\frac{1}{n}$	$\frac{1}{n}$
Ack / Nack	1	2	1	1	2	1	$2 + log_2(n)$	$4^+ - \frac{1}{n}$	$2 + log_2(n)$

cannot propagate far beyond its source in the network. The only data forwarded unconditionally is S1 packets, which are typically small. Although ALPHA cannot prevent flooding with these path reservation packets, hosts that send large amounts of S1 packets without receiving A1 responses can easily be identified and isolated from the network. As ALPHA-C permits senders to fill S1 packets to their maximum size with pre-signatures, large S1 packets can also be used to waste network resources. Hence, relays should initially limit and later increase the maximum size of S1 packets per sender to combat floods of large unsolicited S1 packets.

For incremental deployment, end hosts using ALPHA do not require all or any relays to use it, too. Moreover, even isolated ALPHA-enabled relays can perform per-packet authentication in the network. This eases the deployment in networks with long-lived hardware because ALPHA-capable devices can be added incrementally.

The ALPHA base protocol does not require relays to interact beyond the forwarding packets (e.g., for sharing symmetric keys, synchronizing clocks, etc.). Therefore, it does not introduce new vulnerabilities from malicious relays because it does not rely on the distinction between outsiders and insiders. This makes ALPHA particularly suited for scenarios with dynamic membership in which no pre-shared secrets or distinct roles can be assumed. ALPHA can secure communication between arbitrary nodes in WSN and is not restricted to communication towards a fixed sink or base station. Hence, it can protect WSN applications that require end-to end and on-path integrity-protection for any pair of nodes.

Compared to traditional symmetric and asymmetric signatures, ALPHA signatures exhibit a larger delay in communication due to the additional RTT for delivering the S1 and A1 packet. Thus, their applicability depends on the end-to-end latency of the network and the maximum acceptable delay at the application layer. For scenarios in which the maximum acceptable latency is below 1.5 RTTs, ALPHA signatures are not applicable. ALPHA depends on the stability of the routing path for a minimum of 2 RTTs plus the time for an optional handshake protocol for bootstrapping. With this stability, the minimum amount of packets necessary to transfer payload (S1&2 and A1&2) can traverse the same path. In ALPHA-C and ALPHA-M, the necessary period of stability extends to all packets belonging to the same signature process. In particular, the number of parallel transmissions and the size of the MT should be adapted to the network dynamics for best performance. Additional latency for payload transmission can be introduced by packet loss. Especially S1 and A1 packets require robust and fast retransmission.

Finally, ALPHA is not a complete security solution but must be supplemented with additional components, such as a handshake procedure (cf. Section 3.4) and bypass protection (cf. Section 3.1.1). Specifying these mechanisms is out of scope for ALPHA because they need to meet the particular demands and capabilities of each specific application scenario.

4. EVALUATION

The multi-hop networks for which we propose ALPHA signatures are MANETs, WMNs, and WSNs, where CPU, memory, and energy resources are typically scarce. Thus, in this section, we examine the performance and applicability of our approach with a focus on those three application areas.

Table 1 analyzes the computational costs of the the three ALPHA modes. Asterisks (*) indicate MAC computations on the (variable) sizes of protected messages. All other hash operations are performed on fixed-length input data of the size of one or two hash outputs. Entries marked with a cross ($^+$) are not directly tied to packet handling and can be computed off-line, e.g., in phases of low CPU load, to reduce response time and level CPU load peaks. Tables 2 and 3 compare the buffering-related memory requirements of ALPHA, ALPHA-C, and ALPHA-M. The tables show that only little data needs to be stored on relaying nodes, making resource exhaustion attacks on the path nodes more difficult. Functionally, the integrated support for acknowledgments in ALPHA signatures avoids an additional three-way signature process that interactive hash chain signatures would require for the same functionality. Also, the option for parallelizing transfers via

Table 2: Memory requirements for n messages sent in parallel. (message size: m, hash size: h)

	Signer	Verifier	Relay
ALPHA	$n(m+h)$	$n \cdot h$	$n \cdot h$
ALPHA-C	$n(m+h)$	$n \cdot h$	$n \cdot h$
ALPHA-M	$n \cdot m + (2n-1)h$	h	h

Table 3: Additional memory requirements for n parallel acknowledgedements. (hash size: h)

	Signer	Verifier	Relay
ALPHA	$2n \cdot h$	$2n \cdot h$	$2n \cdot h$
ALPHA-C	$2n \cdot h$	$2n \cdot h$	$2n \cdot h$
ALPHA-M	h	$n \cdot s + (4n-1)h$	h

multiple outstanding packets with the ALPHA-C and ALPHA-M variants significantly increases the bandwidth available to applications. Cumulative transmissions and signed MTs in particular permit a dynamically tunable tradeoff between memory and CPU requirements, latency, and throughput. With this flexibility, ALPHA signatures can adapt to both infrequent low-volume signaling traffic and high-volume data transfers, including changes of a data flow between different traffic patterns. Thus, forged and modified packets can be detected early and flooding-based DoS attacks can be mitigated effectively.

4.1 Application Scenario-specific Evaluation

In the following, we evaluate the feasibility of ALPHA and its variants for three different application scenarios and platforms. Firstly, we evaluate the performance on lightweight mobile devices and workstations. Secondly, we consider the use of ALPHA in less ressource-constrained wireless multi-hop scenarios, such as WMNs that require high throughput. Finally, we evaluate the use of ALPHA in a sensor network scenario with tightly resource-constrained sensor nodes with a characteristically small packet payload.

4.1.1 Performance on Lightweight Mobile Devices

We implemented the ALPHA signature scheme as a lightweight integrity protection scheme for securely signaling association-relevant information to end-hosts and middleboxes. We integrated ALPHA as lightweight security layer for signaling traffic into the Host Identity Protocol (HIP) [16] to show the feasibility of extending existing protocols with ALPHA. In this context, we measured the performance of ALPHA signatures between a Nokia 770 Internet Tablet with a 220 MHz ARM-926 CPU and a server with an Intel Xeon 3.2 GHz CPU. Table 4 lists the performance of the ALPHA signature steps as the mean results of 300 signatures.

The values include the time for packet creation and packet handling (e.g. context switches for user-space processing, de-multiplexing, packet parsing, extraction of packet parameters, etc.), and thus, reflect the actual performance of ALPHA-enabled HIP on the evaluated systems. We also provide the results for RSA and DSA signatures, as used by HIP, although we cannot stress enough that the security properties of ALPHA are not directly comparable to the security properties of PK signatures. We also provide the computation time for a single hash SHA-1 hash function to show the influence of the hash computation on the total processing time.

The measurements show that the overall overhead for processing the ALPHA signature is 2.34 ms if the N770 acts as sender and 3.07 ms if it acts as verifier. Hence, ALPHA signatures significantly reduce the computational cost of signaling with HIP.

Table 4: ALPHA, RSA and DSA delay

	Nokia 770	Xeon 3.2GHz
Send S1	0.33 ms	0.03 ms
Process S1, send A1	1.47 ms	0.05 ms
Process A1, send S2	1.52 ms	0.05 ms
Verify S2, send A2	1.60 ms	0.05 ms
Process A2	0.49 ms	0.05 ms
Sender(total)	2.34 ms	0.13 ms
Receiver (total)	3.07 ms	0.10 ms
SHA-1 Hash	0.02 ms	0.01 ms
RSA 1024 sign	181.32 ms	9.09 ms
RSA 1024 verify	10.53 ms	0.15 ms
DSA 1024 sign	96.71 ms	1.34 ms
DSA 1024 verify	118.73 ms	1.61 ms

Table 5: SHA-1 delay on wireless routers

	AR2315	Broadcom 5365	Geode LX
20 Byte digest	0.059 ms	0.046 ms	0.011 ms
1024 Byte digest	0.360 ms	0.361 ms	0.062 ms

4.1.2 Performance Estimation for WMNs

Besides the latency and computational cost on end-hosts, the computational cost for forwarding nodes is of particular importance because it limits the verifiable throughput in multi-hop scenarios and introduces additional end-to-end latency. In this section, we evaluate ALPHA in a WMN scenario and show the impact of the cryptographic operations on wireless mesh routers. The performance estimates in this section base on two pieces of commodity hardware and one custom-built mesh router: The “La Fonera” wireless router with a 180 MHz Atheros AR2315 32-bit MIPS CPU and the Netgear WGT634U with a 200 MHz Broadcom 5365 MIPS-32 based CPU are widely used in private and unmanaged WMNs. Additionally we consider a customized mesh router with a 32-bit 500 MHz AMD Geode LX800 x86 processor as hardware platform for managed WMNs. Note that the following approximations assume the CPU to be available exclusively for cryptography operations and do not reflect complete packet processing costs as it largely depends on the specific use-case and environment.

The measured computational cost of creating SHA-1 digests for the three devices is given in Table 5. For evaluating ALPHA-C performance, we chose a payload size of 1024 B per packet and cumulative transmissions with 20 pre-signatures per S1 packet, which results in an upper bound for payload throughput of about 20 Mbit per second for both commodity hardware devices. The upper bound for the AMD Geode mesh node is approx-

Table 6: Estimates for ALPHA-M

Leaves	Processing (μs)		Payload (byte)	Throughput (Mbit/s)		Data per S1 (Mbit)
	AR	Geode		AR	Geode	
16	599	258	924	11.8	27.3	0.1
32	660	320	904	10.4	21.5	0.2
64	718	382	884	9.4	17.7	0.4
128	778	444	864	8.5	14.8	0.8
256	837	505	844	7.7	12.7	1.6
512	897	567	824	7.0	11.1	3.2
1024	956	629	804	6.4	9.8	6.3

imately 120 Mbit/s. In these results, the computation of the SHA-1 MAC is responsible for 99% of the total computational cost, dwarfing the cost of verifying the hash chain element in the S1 packet, i.e., the effective overhead introduced by the ALPHA signatures.

Table 6 provides estimates for the computational overhead of ALPHA-M. The throughput refers to the upper bound of data verifiable by the AR2315 and the AMD Geode CPU. The increasing number of S2 packets per S1 results in an increased number of MT leaves, and thus, in less payload due to the larger signature consisting of more MT nodes. Processing time per packet also increases with the number of MT nodes in each packet. However, the exponential growth of the number of parallel S2 transmissions per S1 helps the signer to achieve a better adaptation to its available bandwidth, albeit at an increased computational cost. The larger number of parallel S2 transmission results in a larger amount of signed data per S1, permitting to send fewer S1s or more data in a given time span. Therefore, ALPHA-M in combination with ALPHA-C enables a fine-grained adaptation to network bandwidth, buffer-space, and computational capabilities.

4.1.3 *Performance Estimation for WSNs*

Key factors influencing hop-by-hop authentication performance in WSNs are the limited CPU and memory resources of relaying sensor nodes and the small packet payload. We focus on ALPHA-C because ALPHA-M suffers in throughput from the small packet sizes prevalent in WSNs (cf. Figure 5). As in the previous section, only cryptographic CPU load is taken into account.

We used the Matyas-Meyer-Oseas (MMO) hash function [13] and measured its performance on the Aquis-Grain 2.0, a node with 8 KBs of RAM and a 16 MHz CC2430 system-on-chip. For computing the MMO hash function, we utilize the built-in AES-128 hardware support of the CC2430. Applying the hash function to a 16 byte input string consumes 0.78 ms and 2.01 ms for a 84 byte input. These measurements include the time necessary for transferring the data between the node's memory and the network chip. We base our estimations on these measurements, the values from Table 1, and an assumed packet payload of 100 B[5]. Additionally, we use ALPHA-C with 5 pre-signed messages per S1. Based on these assumptions, we provide an example computation but, depending on the target scenario, smaller packets or hashes can be used due to the resource-constraints of WSNs. The signature overhead per packet is 16 B for the hash chain element of the signer, 16 B for the MAC, and $\frac{16}{5}$ B for the pre-signature in the S1 packet. In this scenario, relays are estimated to verify up to 244 Kbit/s signed payload in 460 S2 packets, being close to the maximum theoretical IEEE 802.15.4 throughput of 250 Kbit/s and well above the practically achievable throughput in real IEEE 802.15.4 networks. The use of pre-acks reduces the maximum amount of verifiable payload data to 156.56 Kbit/s in a total of 334 packets. By utilizing the same algorithms and hardware acceleration features as traditional symmetric-key point-to-point integrity protection approaches, ALPHA achieves a similar performance. However, symmetric cryptography does not allow relays to verify the authenticity of packets, and hence, cannot prevent forged packets from being forwarded.

[5]For IEEE 802.15.4, the packet payload ranges from 80 to 116 B, depending on the applied security measures.

In comparison, public key cryptography in WSNs, even efficient ECC implementations, perform poorly when compared to ALPHA. According to Gura [7] an 160-ECC point multiplication takes 0.81 seconds on an 8 MHz Atmega 128 CPU. Thus, purely ECC-based mechanisms for signature and verification would cause unacceptably high delays for on-path verification. However, ECC signatures present a viable solution for securely exchanging the anchors of hash chains in the beginning of an association (cf. Section 3.4).

5. SUMMARY

Our Adaptive and Lightweight Protocol for Hop-by-hop Authentication (ALPHA) enables efficient end-to-end and hop-by-hop authentication in multi-hop networks. Forwarding nodes can efficiently verify ALPHA-protected data on a per-packet basis, enabling early detection of unsolicited or forged packets. Hence, it can efficiently mitigate flooding attacks and provides a basis for secure signaling to middleboxes.

ALPHA comprises three different modes of operation, namely the base protocol, ALPHA-C with cumulative transmissions, and ALPHA-M combined with MTs. These modes complement each other and allow for a fine-grained and dynamic adaptation to different communication scenarios ranging from transmission of infrequent control traffic to sending large amounts of data. Moreover, the three modes can be combined to fine-tune the performance of ALPHA, and thus, meet the networking, buffering, and computing capabilities of a wide range of device classes. ALPHA natively supports acknowledged as well as unacknowledged packet delivery in all of its modes, making ALPHA suitable for many applications.

6. CONCLUSION

Our performance analysis shows that ALPHA scales and integrates well with different multi-hop networks and application scenarios. The adaptive nature and distinctive resource efficiency enables on-path verification of all data traffic, and thus a much higher level of security in resource-constrained wireless networks. With ALPHA as the foundation, wireless networks can support efficient and secure signaling paths to all network nodes and they become considerably more robust against flooding attacks from outsider as well as from insiders. Being incrementally deployable, ALPHA can improve the security even in heterogeneous networks consisting of ALPHA-aware and unaware relays. Thus, we believe that ALPHA is a valuable authentication scheme for protocol development in wireless and wired multi-hop networks. It provides an elegant, flexible, and efficient alternative to public-key based and symmetric integrity protection, enriching the set of security mechanisms for ubiquitous mobile communication.

7. ACKNOWLEDGMENTS

We want to thank Mark Sillner and Elias Weingärtner for contributing to the quality of this paper.

8. REFERENCES

[1] Akbani, R., Korkmaz, T., and Raju, G. HEAP: Hop-by-hop Efficient Authentication Protocol For Mobile Ad-hoc Networks. *CNS 07* (2007).

[2] Anderson, R., Bergadano, F., Crispo, B., Lee, J., Manifavas, C., and Needham, R. A new family of authentication protocols. *Operating systems review 32*, 4 (1998).

[3] Bellare, M., Canetti, R., and Krawczyk, H. Keying hash functions for message authentication. *CRYPTO 96* (1996).

[4] Bergadano, F., Cavagnino, D., and Crispo, B. Chained Stream Authentication. *Selected Areas in Cryptography: 7th Annual International Workshop, SAC 2000* (2000).

[5] Cheung, S. An efficient message authentication scheme for link state routing. *13th Annual Computer Security Applications Conference* (1997).

[6] Gouda, M. G., Elnozahy, E. N., Huang, C.-T., and McGuire, T. M. Hop integrity in computer networks. *IEEE/ACM Transactions on Networking 10* (2002).

[7] Gura, N., Patel, A., Wander, A., Eberle, H., and Shantz, S. Comparing Elliptic Curve Cryptography and RSA on 8-Bit CPUs. *Cryptographic Hardware and Embedded Systems: 6th International Workshop* (2004).

[8] Hauser, R., Przygienda, A., and Tsudik, G. Reducing the Cost of Security in Link State Routing. *NDSS '97* (1997).

[9] Hu, Y.-C., Perrig, A., and Johnson, D. Packet leashes: a defense against wormhole attacks in wireless networks. *INFOCOM* (2003).

[10] Lamport, L. Password authentication with insecure communication. *Commun. ACM*, 11 (1981).

[11] Liu, D., Ning, P., Zhu, S., and Jajodia, S. Practical broadcast authentication in sensor networks. *MobiQuitous* (2005).

[12] Lu, B., and Pooch, U. W. A Light-weight Hop-by-hop Authentication Protocol for Mobile Ad Hoc Networks. *International Journal of Information Technology 11*, 2 (2005).

[13] Matyas, S., Meyer, C., and Oseas, J. Generating strong one-way functions with cryptographic algorithm. *IBM Technical Disclosure Bulletin 27*, 10A (1985).

[14] Merkle, R. C. A digital signature based on a conventional encryption function. In *CRYPTO '87* (1988).

[15] Merkle, R. C. A certified digital signature. In *CRYPTO '89* (1990).

[16] Moskowitz, R., Nikander, P., Jokela, P., and Henderson, T. Host identity protocol. RFC 5201, IETF, 2008.

[17] Perrig, A., Canetti, R., Song, D., and Tygar, J. Efficient and Secure Source Authentication for Multicast. *NDSS '01* (2001).

[18] Perrig, A., Canetti, R., Tygar, D., and Song, D. The TESLA broadcast authentication protocol. *Cryptobytes 5* (2002).

[19] Torvinen, V., and Ylitalo, J. Weak Context Establishment Procedure for Mobility Management and Multi-Homing. *IFIP Conference on Communications and Multimedia Security* (2004).

[20] Weimerskirch, A., and Westhoff, D. Zero Common-Knowledge Authentication for Pervasive Networks. *SAC '03* (2003).

[21] Yao, T., Fukunaga, S., and Nakai, T. Reliable broadcast message authentication in wireless sensor networks. In *EUC Workshops* (2006).

[22] Ye, F., Luo, H., Lu, S., and Zhang, L. Statistical En-Route Filtering of Injected False Data in Sensor Networks. *IEEE Journal on Selected Areas in Communications 23* (2005).

[23] Zhang, K. Efficient protocols for signing routing messages. *NDSS '98* (1998).

[24] Zhang, W., Subramanian, N., and Wang, G. Lightweight and compromise-resilient message authentication in sensor networks. *INFOCOM 2008* (2008).

[25] Zhu, S., Setia, S., Jajodia, S., and Ning, P. An interleaved hop-by-hop authentication scheme for filtering of injected false data in sensor networks. *Security and Privacy* (2004).

[26] Zhu, S., Xu, S., Setia, S., and Jajodia, S. LHAP: a lightweight hop-by-hop authentication protocol for ad-hoc networks. *Distributed Computing Systems Workshops, 2003* (2003).