Cumulative Message Authentication Codes for Resource-Constrained Networks

He Li, Vireshwar Kumar, Jung-Min (Jerry) Park, and Yaling Yang
Department of Electrical and Computer Engineering, Virginia Tech, USA
{heli, viresh, jungmin, yyang8}@vt.edu

***Abstract*—In emerging applications, such as intelligent automotive systems, Internet-of-Things (IoT) and industrial control systems, the use of conventional message authentication codes (MACs) to provide message authentication and integrity is not possible due to the large size of the MAC output. A straightforward yet naive solution to this problem is to employ a truncated MAC which undesirably sacrifices cryptographic strength in exchange for reduced communication overhead. In this paper, we address this problem by proposing a novel approach for message authentication called *Cumulative Message Authentication Code* (CuMAC), which consists of two distinctive procedures: *aggregation* and *accumulation*. In aggregation, a sender generates compact authentication tags from segments of multiple MACs by using a systematic encoding procedure. In accumulation, a receiver accumulates the cryptographic strength by verifying the authentication tags and collecting the underlying MAC segments encoded in them. Embodied with these two procedures, CuMAC enables the receiver to achieve an advantageous trade-off between the cryptographic strength and the latency in processing of the authentication tags. We have carried out comprehensive evaluations of CuMAC in two real-world applications: low-power wide-area network and in-vehicle controller area network. Our evaluation methodology included simulations as well as a prototype implementation of CuMAC on a real car.**

***Index Terms*—Message authentication code (MAC); Internet-of-Things (IoT); Sigfox; Controller area network (CAN).**

I. Introduction

In emerging applications, such as home automation, industrial controllers and sensor networks, a large number of energy-constrained computing devices are getting closely integrated with the existing computer infrastructure through bandwidth-constrained networks to form the Internet-of-Things (IoT) [1]. The successful adoption of those applications will partially depend on our ability to thwart security and privacy threats, including message forgery and tampering. Today, message authentication code (MAC) is the most commonly used method for providing message authenticity and integrity in wired/wireless network applications. To employ MACs in a resource-constrained (i.e., energy and/or bandwidth constrained) network, we need to consider two problems: the computational burden on the devices for generating and verifying the MAC, and the additional communication overhead incurred due to the inclusion of the MAC in each message frame/packet. The first problem can be addressed by using dedicated hardware and cryptographic accelerators [2], [3]. However, the second problem is not as easy to address.

The cryptographic strength of a MAC depends on the cryptographic strength of the underlying cryptographic primitive (e.g., a hash or block cipher), the size of the MAC output, and the size and quality of the key. Hence, a *conventional MAC* scheme typically employs at least a few hundred bits of MAC output to ensure a sufficient level of cryptographic strength. In energy-constrained networks (e.g., low-power wide-area network with battery-powered nodes) and bandwidth-constrained networks (e.g., in-vehicle network), the payload size of each packet is very short, e.g., less than 150 bits in protocols like Sigfox [1] and controller area network [4]. Hence, not more than a few bits of the payload can be spared to include an *authentication tag* associated with the MAC.

The legacy solution for generating a short authentication tag is to truncate the output of a conventional MAC so that it fits a message packet [5]–[7]. This type of MAC is called a *truncated MAC*. However, in exchange for reduced communication overhead and energy consumption, the truncated MAC sacrifices cryptographic strength which may be undesirable, or even unacceptable, in some applications. Note that the truncated MAC without sufficient cryptographic strength renders the application vulnerable to collision attacks [8]. To enable authentication with enhanced cryptographic strength, Katz et al. propose the concept of *aggregate MAC* where conventional MACs of multiple messages are combined into one aggregate MAC, and transmitted over successive packets [9]. Similarly, Nilson et al. propose a *compound MAC* which is calculated on a compound of multiple messages, and distributed over successive packets [4]. However, both the aggregate and compound MAC schemes incur significant latency in the verification of the messages because the receiver needs to receive and process all associated packets before being able to verify the MAC.

In summary, we identify two challenges in employing MACs for resource-constrained networks: (1) incurring minimal communication overhead so that the MAC can fit in a message packet, and (2) ensuring that the cryptographic strength meets the security need of the application. In this paper, we propose a novel approach for message authentication that we refer to as *Cumulative Message Authentication Code* (CuMAC) that addresses both of the aforementioned challenges. In CuMAC, a sender utilizes a procedure called *aggregation* through which the sender first divides the full-sized MAC output of each message into multiple short MAC segments, and then "aggregates" the MAC segments of multiple messages using a systematic encoding procedure to form a short authentication tag. This procedure resolves the first challenge of ensuring low communication overhead.

Further, the receiver utilizes a procedure called *accumulation* through which it first verifies the MAC segments aggregated into the authentication tag of each received packet, and then "accumulates" the cryptographic strength by collecting the verified MAC segments associated with the target message. In this procedure, the receiver may incur delay that is proportional to the accumulated cryptographic strength since it needs to wait for the relevant tags to be received and processed. Hence, while the accumulation procedure caters to the second challenge, it brings up a novel trade-off between the cryptographic strength and delay. CuMAC enables the receiver to authenticate the message in real-time with the cryptographic strength which is commensurate with the size of each tag. Further, CuMAC enables the authentication with the highest level of cryptographic strength (which is commensurate with size of the MAC) after accumulating all segments of the MAC that covers the message in the associated packets.

The paper's main contributions are summarized as follows.

1) We propose a novel message authentication scheme called *CuMAC* which meets the security need of resource-constrained networks. CuMAC is an embodiment of two concepts that we refer to as *aggregation* (which reduces the communication overhead) and *accumulation* (which increases the cryptographic strength).
2) We have thoroughly evaluated the effectiveness of CuMAC through simulations in energy-constrained and bandwidth-constrained networks. Our results illustrate that while incurring the same communication overhead as the truncated MAC scheme, CuMAC achieves the cryptographic strength equivalent to the conventional MAC scheme at the cost of increase in latency.
3) We validate our analytical and simulation results using a prototype implementation on a real car.

II. Potential Application Scenarios

We discuss two suitable application scenarios of CuMAC, where the constraints of the network—either in terms of MAC size or energy/bandwidth consumption of the networked devices—prohibit the use of the conventional MAC scheme. We note that the design of CuMAC is *not* limited by specific characteristics of these two applications. As such, CuMAC can be readily employed in a variety of other IoT networks [10]–[12] satisfying the system model discussed in Section III-A.

A. Low-Power Wide-Area Network (LPWAN)

Many IoT applications (e.g., smart metering and smart city infrastructure) require a heavily-crowded network of low-cost energy-constrained battery-operated wireless devices. The paradigm of LPWAN is aimed at fulfilling these requirements of IoT networks [1], [5]. Sigfox [13] is one example of a widely-known LPWAN technology. In Sigfox, each uplink packet contains a counter, a message (with length between 0 and 96 bits), and an authentication tag (with length between 16 and 40 bits). To enable robust communication over the unreliable wireless channel, the sender in Sigfox transmits multiple copies of the same packet sequentially. After transmitting the fixed number of copies of the packet, Sigfox waits for an acknowledgement from the receiver. In the absence of the acknowledgement, the packet is considered lost. We note that Sigfox does not support retransmission of lost packets.

The battery-powered Sigfox devices are expected to have a service/battery life of several years. As the energy consumption of a Sigfox device is directly proportional to the size of packet communicated by it, it is imperative to communicate using short packets to ensure a long battery life. Also, although the message integrity and authentication are of prime importance in applications supported by Sigfox [14], it is unfeasible to communicate the full-sized MAC output due to the small size of the tag allocated in the Sigfox packet.

B. In-Vehicle Controller Area Network (CAN)

Today's high-end cars use a hundred or more electronic control units (ECUs) to enable advanced functionalities, such as real-time engine control. ECUs in most modern vehicles communicate with each other over a bandwidth-constrained wired broadcast channel called the Controller Area Network (CAN) bus [15], [16]. Because the messages communicated among ECUs directly affect vital functions of a vehicle, some of which are safety related (e.g., dynamics control system [17]), the security and reliability of the CAN bus and the integrity of the messages on it are critical [3]. We note that while the state-of-the-art CAN bus supports robust mechanisms for message acknowledgement and retransmission of corrupted/lost packets, it does not support any security mechanism [18]. Several studies have shown that a car's in-vehicle network can be compromised through either direct physical access (e.g., using the on-board diagnostics port) or a remote connection (e.g., using Bluetooth) to the CAN bus [19], [20]. Due to one such vulnerability, Jeep had to recall 1.4 million vehicles in 2015 [21]. To counter such attacks and protect messages on the CAN bus, the US National Highway Traffic Safety Administration (NHTSA) recommends the inclusion of MACs [22].

A CAN packet consists of an 11-bit or a 29-bit identifier field and a message field with length between 0 and 64 bits. Except the identifier and message fields, we cannot arbitrarily change the length or the content of other fields in the CAN packet as that would make the modified packet incompatible with the existing CAN protocol. Hence, in the prior art [6], [23], to realize MAC-based authentication in each packet, the identifier field is used to accommodate an 18-bit counter, and the message field is used to accommodate the message payload as well as the authentication tag. We note although the design of this modified packet ensures that it is backward-compatible, inserting a full-sized MAC in the modified packet is not possible because the maximum allowed length of the message field in a CAN packet is only 64 bits.

Fig. 1: Packet model employed in CuMAC.

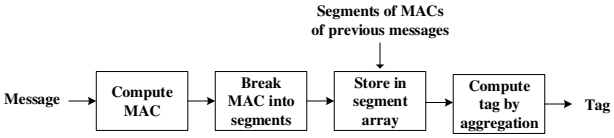


Fig. 2: Schematic of the procedures in the tag generation algorithm at the sender in CuMAC.

III. Overview of CuMAC

A. System Model

We consider an energy-constrained (e.g., Sigfox) and/or bandwidth-constrained network (e.g., CAN) where a sender needs to transmit security-critical messages to a receiver using small packets. As shown in Figure 1, we let the sender employ a packet format which contains at least three fields: a packet counter, a message, and an authentication tag. We note that these three fields are critical for ensuring any secure message authentication scheme including CuMAC. Hence, if the network protocol (e.g., Sigfox as discussed in Section II-A) employs these fields in the conventional packets by design, CuMAC can readily utilize them; otherwise, the packet contents can be modified in the target network protocol (e.g., CAN as discussed in Section II-B) to include these fields.

We assume that there exists a message acknowledgement mechanism which enables the sender to know if a particular packet was correctly delivered to the receiver [24]. Such acknowledgement mechanisms are widely utilized in existing protocols including Sigfox and CAN. The acknowledgement mechanism assisted with the packet counter enables the sender and the receiver to maintain the same sequence of packets. Note that we do not make any assumption about the message retransmission mechanism, i.e., the network may or may not support retransmission. We highlight that in this paper, we provide Sigfox and CAN as concrete application scenarios for CuMAC, but our system model is generically applicable to a variety of resource-constrained networks, e.g., those employing Bluetooth Low Energy [10], Constrained Access Protocol [11] or Message Queue Telemetry Transport [12].

B. Design of CuMAC

In the above system model, the sender and the receiver (after sharing a secret key) communicate a sequence of messages and employ CuMAC for authentication. CuMAC comprises of two major algorithms: tag generation and tag verification. In the tag generation algorithm, the sender computes the authentication tag through two major steps (Figure 2). In the first step, the sender generates the MAC of the message, breaks the MAC into n short segments, and stores them into a segment array. In the second step, the sender retrieves n segments (one MAC segment of the current message, and $n-1$ segments of the MACs of the previously transmitted messages) from the segment array, and aggregates the segments to generate a tag.

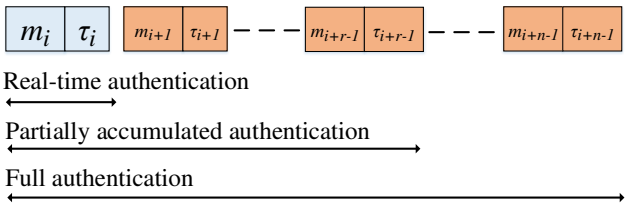


Fig. 3: Illustration of the levels of authentication in CuMAC.

For instance, in the illustration shown in Figure 3, an L-bit MAC is divided into n segments, such that the size of each segment is l bits, i.e., $L = n \cdot l$. Then, an authentication tag of length l bits is computed using the tag generation algorithm of CuMAC (Figure 2). Finally, in the i^{th} packet, the sender transmits the message denoted by m_i and the authentication tag denoted by τ_i. We note that in CuMAC, the n segments of the MAC of the message m_i are aggregated into the n authentication tags, $\tau_i, \cdots, \tau_{i+n-1}$, and transmitted in the corresponding packets.

Having received each packet, the receiver runs the tag verification algorithm which includes two major steps. In the first step, the receiver generates an authentication tag of the received message using the same procedure employed in the tag generation algorithm. In the second step, the receiver compares the generated authentication tag with the received authentication tag. If the authentication tags match, the receiver accumulates the MAC segments (aggregated in the authentication tag) with the previously received MAC segments of the corresponding message. For instance, in the illustration shown in Figure 3, after receiving and verifying each of the n authentication tags, $\tau_i, \cdots, \tau_{i+n-1}$, the receiver accumulates the MAC segments of the message m_i (aggregated in those tags) to reconstruct the underlying L-bit MAC.

C. Authentication Levels in CuMAC

For CuMAC, we define three levels/features of authentication: (1) real-time authentication, (2) full authentication, and (3) partially accumulated authentication. Figure 3 illustrates the three different levels of authentication, when applied to message m_i. Recall that the MAC of the message m_i is divided into n segments, and distributed in tags $\tau_i, \cdots, \tau_{i+n-1}$. In this case, the receiver can perform real-time authentication immediately after receiving message m_i by processing the tag τ_i. With real-time authentication, the receiver performs authentication without any delay, but it achieves the lowest cryptographic strength since there is no security accumulation using the subsequent tags. On the other hand, the receiver can perform full authentication after receiving all of the segments of the MAC associated with message m_i in tags $\tau_i, \cdots, \tau_{i+n-1}$. With full authentication, the receiver achieves the highest cryptographic strength, but needs to incur a latency of $n-1$ packets. The receiver can perform partially

accumulated authentication by accumulating and processing tags $\tau_i, \cdots, \tau_{i+r-1}$, where $1 < r < n$. Partially accumulated authentication enables the receiver to make a trade-off between cryptographic strength and message verification latency to meet the security and performance needs of the application.

D. Attack Model

We consider an adversary which aims to forge valid authentication tags for its malicious messages so that it can deceive the authentication scheme at the receiver. Specifically, to break the real-time authentication feature of CuMAC, the adversary needs to forge a message and a valid tag. The forgery need to be *fresh* which means that the sender has not transmitted the MAC of the same counter and message pair. To break the partially accumulated authentication feature of CuMAC with r accumulated segments, the adversary need to forge a sequence of r messages with valid tags. In this sequence, the forgery for only the first message needs to be fresh. Similarly, to break the full authentication feature of CuMAC, the adversary need to forge a sequence of n messages with valid tags, where forgery for at least the first message is fresh. While the adversary can eavesdrop the communication channel to obtain packets transmitted by the sender, it does not know the secret key which is shared between the sender and the receiver, and utilized for generating/verifying the authentication tags.

E. Security Objectives

We convey the cryptographic strength in bits, where a cryptographic strength of λ bits for a scheme means that for any adversary making at most 2^λ queries or taking at most 2^λ time, the probability of successfully launching an attack on the scheme is negligibly small [25]. The cryptographic strength of a conventional MAC depends on three security parameters: (1) the cryptographic strength of the underlying cryptographic primitive, (2) the size and quality of the secret key, and (3) the size of the MAC output. To achieve a cryptographic strength of λ bits, the minimum size of the key and the MAC output should be λ bits. In this paper, we present the cryptographic strength using the size of the MAC output (denoted by L).

From the illustration discussed in Section III-C, we note that the cryptographic strength of the full authentication depends on the same three aforementioned security parameters of the conventional MAC. However, the cryptographic strength of real-time authentication in CuMAC is limited by the size of the MAC segment l. Also, the cryptographic strength of the partially accumulated authentication in CuMAC depends on the size of the MAC segment l and the number of accumulated segments r. The security objective of CuMAC is to ensure that the probability with which an adversary succeeds in breaking each of the three authentication features is negligible (i.e, commensurate with the corresponding cryptographic strength).

Due to space constraints, a formal discussion of CuMAC's security properties and associated proofs are provided in the extended version of this paper, which is available at [26].

IV. Technical Details of CuMAC

Here, we present the technical details of the algorithms employed by CuMAC. We also provide an example that illustrates the generation and verification of the tags in CuMAC.

A. Algorithms

CuMAC is composed of the following algorithms that are executed by the sender and/or the receiver.

$\mathtt{k} \leftarrow \mathsf{KeyGen}(1^\lambda)$

This probabilistic key generation algorithm is utilized by the sender and the receiver to obtain the secret key. The input to this algorithm is the security parameter $\lambda \in \mathbb{N}$, and the output is the secret key denoted by $\mathtt{k}$. In a resource-constrained network, this algorithm can be efficiently realized by leveraging a trusted third party [7], or using an efficient key distribution mechanism [27].

$\sigma_i \leftarrow \mathsf{MacGen}(\mathtt{k}, i, m_i)$

This deterministic MAC generation algorithm is utilized by the sender and the receiver (as a sub-algorithm of tag generation and verification algorithms) to compute the MAC of a message using the secret key. The inputs to this algorithm are the secret key $\mathtt{k}$, a counter i and a message m_i. This algorithm outputs the L bits long MAC represented by σ_i. This algorithm can be realized using a cipher-based (e.g., AES-CMAC) or a hash-based (e.g., SHA-3) MAC scheme. In this paper, we utilize the widely used AES-CMAC [28].

$\tau_i \leftarrow \mathsf{SegAgg}(\mathtt{segArray})$

This segment aggregation algorithm is utilized by the sender and the receiver as a sub-algorithm of tag generation and the tag verification algorithms, respectively. It takes as input a two-dimensional array of MAC segments $\mathtt{segArray}$. The i^{th} row of segments in $\mathtt{segArray}$ is generated as follows. The L-bit MAC σ_i is divided into n segments, such that the size of each segment is l bits, i.e., $L = n \cdot l$. The j^{th} segment of σ_i is represented by s_i^j, and is extracted from σ_i as

$$s_i^j \leftarrow (\sigma_i)_{\downarrow[(j-1)\cdot l+1, j\cdot l]}. \tag{1}$$

The notation $\downarrow$ implies that the bits in s_i^j correspond to the bits from $((j-1)\cdot l+1)^{\text{th}}$ bit to $(j\cdot l)^{\text{th}}$ bit in σ_i. Further, this algorithm extracts n elements from $\mathtt{segArray}$ ($n-1$ previous MAC segments and one current MAC segment), and computes the authentication tag τ_i as follows.

$$\tau_i \leftarrow \bigoplus_{j=1, i-j+1>0}^{n} s_{i-j+1}^j. \tag{2}$$

This algorithm outputs the authentication tag τ_i.

$\tau_i \leftarrow \mathsf{TagGen}(\mathtt{k}, i, m_i)$

This tag generation algorithm is run by the sender to generate an authentication tag. It takes as inputs the secret key $\mathtt{k}$, a counter i and a message m_i. It utilizes an array of MAC segments $\mathtt{segTx}$ which is stored and maintained by the sender. This algorithm proceeds as follows to output the authentication tag τ_i.

1) Compute the MAC of the message m_i and set it as σ_i, i.e., $\sigma_i \leftarrow \mathsf{MacGen}(\mathtt{k}, i, m_i)$.
2) Divide the MAC σ_i into n segments as shown in equation (1) and append the segments to the array `segTx`.
3) Compute and output the tag τ_i by aggregating the segments of MACs in `segTx` as shown in equation (2), i.e., $\tau_i \leftarrow \mathsf{SegAgg}(\mathtt{segTx})$.

After receiving the positive acknowledgment of the delivery of the packet from the receiver, the sender increments the packet counter i by one for the next packet. We note that the packet counter i can be readily employed to handle the case of a lost packet. The sender gets to know that the i^{th} packet is lost when it does not receive the acknowledgement from the receiver or it receives a negative acknowledgement. In this case, if the sender supports a retransmission mechanism, the sender simply re-transmits the same packet containing the same counter i, the same message m_i and the same tag τ_i. Otherwise, if the sender does not support any retransmission mechanism, the sender does not increment the packet counter, removes the i^{th} row (i.e., the most recently appended row) of segments in `segTx`, and then proceeds with the tag generation of the next message.

`valid`/`invalid` $\leftarrow$ **TagVerify**$(\mathtt{k}, i, m_i, \tau_i)$

This verification algorithm is run by the receiver for verifying the authenticity of the received message and tag. It takes as inputs the secret key `k`, the received counter i, the received message m_i, and the received tag τ_i. It utilizes an array of MAC segments `segRx`, and an array of number of verified segments `accRx`. These arrays are stored and maintained by the receiver. The i^{th} entry in the array `accRx` is represented by r_i. To initialize the value of r_i in the array `accRx`, the receiver sets $r_i = 0$. This algorithm verifies whether the tag τ_i is generated using the secret key `k`. If the verification succeeds, it outputs the value `valid`; otherwise, it outputs the value `invalid`. This algorithm proceeds as follows.

1) Compute the MAC of the message m_i and set it as $\widetilde{\sigma}_i$, i.e., $\widetilde{\sigma}_i \leftarrow \mathsf{MacGen}(\mathtt{k}, i, m_i)$.
2) Divide the MAC $\widetilde{\sigma}_i$ into n segments as shown in equation (1) and append the segments to the array `segRx`. We note that the counter i ensures that the arrays `segTx` at the sender and `segRx` at the receiver remain synchronized.
3) Compute the tag $\widetilde{\tau}_i$ by aggregating the segments of MACs in `segRx` as shown in equation (2), i.e., $\widetilde{\tau}_i \leftarrow \mathsf{SegAgg}(\mathtt{segRx})$.
4) If $\widetilde{\tau}_i = \tau_i$,
   a) Update the array of accumulated MAC segments `accRx`, such that for each $t \in [i-n+1, i]$, set $r_t = r_t + 1$.
   b) Output the value `valid`.
5) Otherwise, if $\widetilde{\tau}_i \neq \tau_i$, output the value `invalid`.

B. Instantiation of CuMAC

Table I presents an example of CuMAC. The size of the tag in each packet is 32 bits (i.e., $l = 32$). The MAC is generated using the AES-CMAC algorithm. Hence, the size of the MAC output is 128 bits (i.e., $L = 128$), which provides cryptographic strength of 128 bits. Each MAC is divided into four segments (i.e., $n = 4$). This means that the cryptographic strengths for real-time authentication and full authentication are 32 bits and 128 bits, respectively. To simplify the discussion, we limit the discussions to the packets which are involved in the authentication of the message transmitted in the fifth packet, m_5. In the fifth packet, the MAC σ_5 of the message m_5 is computed. To compute the corresponding tag τ_5, the sender aggregates the segment s_5^1 of the MAC σ_5 and the segments of the MACs of the previously generated messages, σ_2, σ_3 and σ_4. Further, the tags τ_6, τ_7 and τ_8 are computed using the segments s_5^2, s_5^3 and s_5^4 of σ_5, respectively.

TABLE I: Example illustrating CuMAC with $L = 128$, $n = 4$, and $l = 32$.

Packet Counter	Previous MACs	Current MAC	Aggregation of MAC segments	Tag
5	$\sigma_2, \sigma_3, \sigma_4$	σ_5	$s_2^4 \oplus s_3^3 \oplus s_4^2 \oplus s_5^1$	τ_5
6	$\sigma_3, \sigma_4, \sigma_5$	σ_6	$s_3^4 \oplus s_4^3 \oplus s_5^2 \oplus s_6^1$	τ_6
7	$\sigma_4, \sigma_5, \sigma_6$	σ_7	$s_4^4 \oplus s_5^3 \oplus s_6^2 \oplus s_7^1$	τ_7
8	$\sigma_5, \sigma_6, \sigma_7$	σ_8	$s_5^4 \oplus s_6^3 \oplus s_7^2 \oplus s_8^1$	τ_8

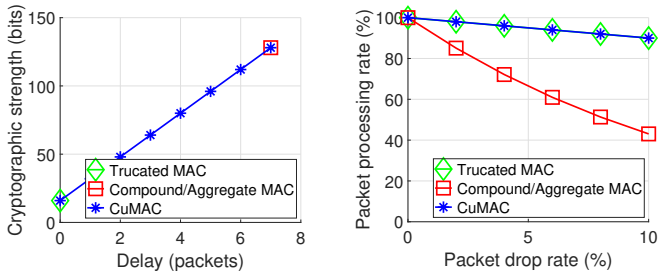
When the receiver receives the fifth packet with the message m_5, the successful verification of the tag τ_5 enables the real-time authentication of message m_5 with the cryptographic strength of 32 bits. Next, the receiver receives and verifies the validity of tags τ_6, τ_7, and τ_8. If all four tags are verified as `valid`, the receiver combines the segments s_5^1, s_5^2, s_5^3 and s_5^4—which are contained in tags τ_5, τ_6, τ_7 and τ_8, respectively—to accumulate the cryptographic strength. This enables the receiver to perform full authentication of message m_5 with the cryptographic strength of 128 $(= 4 \times 32)$ bits. However, if the receiver is restricted to process the fifth packet only after receiving the seventh packet due to latency requirements, it may also perform partially accumulated authentication of message m_5 with a cryptographic strength of 96 bits after verifying tags τ_5, τ_6 and τ_7. Note that this ability to perform the partially accumulated authentication is the most unique feature of CuMAC when compared to the prior art.

V. Evaluation

We firstly highlight the advantages of CuMAC by comparing it with the prior art. We then evaluate the performance of CuMAC in an energy-constrained network application and a bandwidth-constrained network application. These two applications have very different constraints, but they share a common requirement—i.e., messages need to be protected using short tags—that illustrates the utility of CuMAC.

A. Comparison with the Prior Art

We evaluate the performance of CuMAC by comparing it with three other schemes from the prior art: the truncated MAC [6], the compound MAC [4], and the aggregate MAC [9]. For all four schemes, AES-CMAC with a MAC output of 128 bits is utilized as the underlying MAC algorithm. We



(a) Trade-off between cryptographic strength and delay.

(b) Effect of unreliable communication channel.

Fig. 4: Comparison of CuMAC with the prior art.

set the size of the tag in all the four schemes to 16 bits. In the truncated MAC scheme, each MAC is truncated to 16 bits, and transmitted as the tag. In the compound MAC scheme, a compound MAC of 128 bits is computed over eight messages. In the aggregate MAC scheme, an aggregate MAC of 128 bits is computed by aggregating the MACs of eight messages. The compound MAC and the aggregate MAC are divided into eight segments each of size 16 bits, and transmitted in each of the eight packets as the tag. In CuMAC, each MAC of 128 bits is divided into eight segments each of size 16 bits, and each tag is generated by aggregating segments of seven previously transmitted messages and the current message.

Figure 4a presents the cryptographic strengths of four schemes versus their authentication delay. In the figure, we observe that CuMAC provides real-time authentication with cryptographic strength of 16 bits, which is the same as truncated MAC. As more packets are received, partially accumulated authentication is achieved and CuMAC provides gradually increasing cryptographic strength. Finally, CuMAC provides full authentication with cryptographic strength of 128 bits, which is the same as compound/aggregate MAC.

Most importantly, findings shown in Figure 4a highlight one critical advantageous attribute of CuMAC. CuMAC enables a receiver to make a trade-off between (accumulated) cryptographic strength and authentication delay. In some latency-tolerant applications, this attribute provides the receiver with operational flexibility to vary the security level and/or packet processing delay based on particular needs of a protocol or rules prescribed by network traffic processing policies.

Further, we evaluate the effect of unreliable communication channel on the four schemes in Figure 4b. The unreliability of the channel is measured by the packet drop rate which is equal to the ratio of the lost packets and the total number of transmitted packets. The performance of each scheme is measured in terms of the packet processing rate which is equal to the ratio of successfully authenticated packets at the receiver and the total number of transmitted packets. We note that although the packet processing rates in CuMAC and the truncated MAC are equal (Figure 4b), the cryptographic strengths for their full authentication are 128 bits and 16 bits, respectively (Figure 4a).

TABLE II: Parameters utilized for computing the service life of a sensor node in a Sigfox network.

Battery capacity	8000 mAh $\times$ 3600 s/h = 28800 C
Sleep charge	1.3 μA $\times$ 86400 s/day = 0.11 C/day
Packet transmission rate	1 packet/h = 24 packets/day
Packet transmission without payload	0.20 C
Payload transmission	0.002 C/bit

Further, in Figure 4b, we observe that the compound/aggregate MAC can enable processing of significantly lower number of packets than CuMAC. This is because in compound/aggregate MAC, the verification of a MAC requires the receiver to receive *all* of the packets that contain the messages utilized to compute that particular MAC, and loss of any one of those packets leads to the failure in processing of other packets. This implies that given the packet drop rate of ρ, the packet processing rate can be represented by $(1-\rho)$ in CuMAC, but $(1-\rho)^n$ in the compound/aggregate MAC. Hence, for a typical packet drop rate $\rho = 10\%$ and $n = 8$, the packet processing rate in the compound/aggregate MAC is around 43% which is an unacceptable rate in a typical network.

B. Advantages in an Energy-Constrained Network

We consider an air quality monitoring system which consists of a base station and multiple sensors nodes distributed over a large area [13]. Each sensor node utilizes the Sigfox protocol to send the air quality data to the base station once in every hour [29]. The data is examined at the base station and finally made available to the responsible authorities. In this application scenario, there are two important performance requirements—(1) *service life*: each battery-operated sensor node needs to operate for a few years independently without any physical access which means that the network is energy-constrained; and (2) *robust authentication*: message authentication scheme is needed to ensure verification of received data despite losing some packets. Note that in this scenario, the latency requirement is not stringent as the data is collected and analyzed at the base station with some inherent delay.

1) Service Life: We evaluate the effect of appending an authentication tag in each packet on the service life of a sensor node, which is equal to its battery life. To compute the service life, we utilize the charge consumption data from a Sigfox compliant transceiver IC produced by ON Semiconductor [30]. Table II presents the parameters utilized in the computation of the service life. We employ two 1.5 V Alkaline C batteries connected in series. Each battery holds a charge of 8000 mAh. The transmission time in every hour is limited to 2 seconds, and hence the device is considered to be in sleep almost all the time. We ignore the battery self discharge in this calculation.

Figure 5 presents the service life of a sensor node for different sizes of message and authentication tag. We observe that by appending authentication tags in transmitted packets, each sensor node consumes a significantly more energy on data transmission which shortens the service life. Specifically, we consider a 48-bit message without tag as the benchmark which results in the service life of around 11 years. Figure 5

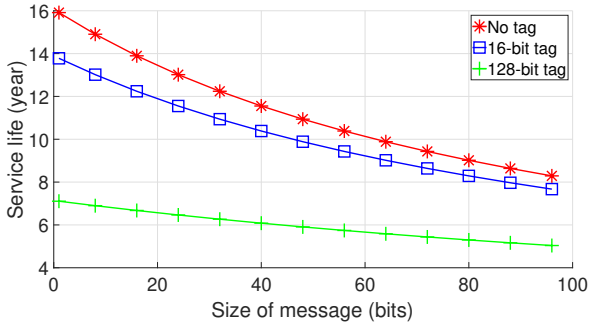


Fig. 5: Effect of size of message and authentication tag on the service life of a sensor node in a Sigfox network.

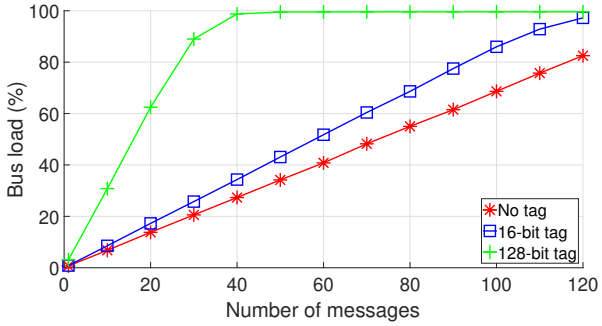


Fig. 6: Effect of size of message and authentication tag on the CAN bus load.

TABLE III: Distribution of size and period of messages in CAN.

Size	1 byte	2 bytes	4 bytes	6 bytes
Share	35 %	49 %	13 %	3 %

Period	5 ms	10 ms	20 ms	50 ms	100 ms	200 ms	1000 ms
Share	7 %	25 %	25 %	3 %	20 %	1 %	19 %

illustrates that in comparison to this benchmark, utilizing the conventional MAC of size 128 bits results in a significant loss of around 45% of service life. However, CuMAC can utilize the 16-bit tag in each packet without compromising the cryptographic strength (128 bits) for full authentication, and with a modest (around 10%) reduction in the service life as compared to the benchmark. Hence, for the LPWAN (like Sigfox) where the size of the tag in each packet is usually limited due to the energy constraints, we assert that CuMAC is a much more viable solution for message authentication than the full-size conventional MAC.

2) Robust Authentication: Recall that in Sigfox, the sender becomes aware of the lost packet when it does not receive the acknowledgement from the receiver. Since Sigfox does not support retransmission of packets, the authentication scheme needs to be robust against packet drops. In CuMAC, the packet counter readily handles such cases, and ensures synchronization of packets between the sender and receiver. This implies that with CuMAC, all received messages can be authenticated with the cryptographic strength of 128 bits, albeit with some delay. However, in this application, the truncated MAC cannot provide high cryptographic strength as shown in Figure 4a, and the compound/aggregate MAC cannot provide robust authentication to all received packets as shown in Figure 4b.

C. Advantages in a Bandwidth-Constrained Network

We consider the CAN bus as an illustrative bandwidth-constrained network. We simulate the performance of the CAN bus when the authentication tag along with the message is inserted in the CAN packets. Table III illustrates the distribution of the size and the period of messages utilized in the simulation. This distribution is based on the open-source benchmark presented by Kramer et al. [31]. The bus speed utilized in the simulation is 500 kbps. In the simulation, we let the maximum size of the message to be 6 bytes which means 2 bytes (16 bits) of tag can be readily inserted in the data field. An 18-bit packet counter is inserted in the CAN identifier field [6], [23]. To communicate the full 128-bit tag, an ECU may employ the *trailing MAC* scheme in which the ECU needs to transmit two extra packets with the tag for each packet with the message [32]. Recall that a maximum of 64 bits of tag can be transmitted in one CAN packet. In the above scenario, we evaluate two performance metrics of the CAN bus: *bus load* and *message processing delay*.

1) Bus Load: The bus load is a critical parameter for evaluating the overall latency performance of the CAN bus. Typically, the CAN bus load is between 30 % to 40 %, but with systematic approaches based on scheduling analysis, the bus load can be increased to around 80 % [33]. The bus load is directly proportional to the number of supported messages on the CAN bus. A high bus load may increase the latency of messages that may lead to problems, such as car functions being delayed and high possibility of communication fault situations [18]. Hence, it is critical to keep the bus load low.

Figure 6 illustrates the effect of increasing the number of messages and inserting authentication tags in the CAN packet on the CAN bus load. We observe that at a typical bus load of 40 %, the number of supported messages without authentication is 60. While maintaining the same bus load, CuMAC with a 16-bit tag is able to support 45 messages, but a trailing MAC with a 128-bit tag supports only 12 messages. Further, considering the maximum bus load of 80 %, the maximum number of messages supported by the bus with a 16-bit tag is 91, but that with a 128-bit tag is only 27. This means that to support 91 messages, a vehicle needs only one CAN bus when the messages are authenticated using CuMAC, but it needs three CAN buses when the messages are authenticated using a full-size MAC. Note that increasing the number of CAN buses increases the overall cost of the vehicle.

2) Message Processing Delay: Message processing delay is an important design metric for the CAN bus which supports safety-critical functions of a vehicle [33]. The major components of this delay includes the delay in the generation of the authentication tag at the sender, communication of the CAN packet over the bus, and then verification of the authentication

TABLE IV: Comparison of the MAC schemes using the prototype implementation on a real car.

Scheme	Code Space	Increase in Bus Load	Real-Time Auth.		Full Auth.		Partially Accum. Auth.	
			Delay	Strength	Delay	Strength	Delay	Strength
Trailing MAC	7410 bytes	300 %	3.451 ms	0 bit	5.616 ms	128 bits	50.000 ms	128 bits
Truncated MAC	7410 bytes	8 %	3.440 ms	16 bits	3.440 ms	16 bits	50.000 ms	16 bits
Compound/Aggregate MAC	7450 bytes	8 %	3.887 ms	0 bit	84.143 ms	128 bits	50.000 ms	0 bits
CuMAC	7522 bytes	8 %	3.798 ms	16 bits	83.983 ms	128 bits	50.000 ms	64 bits

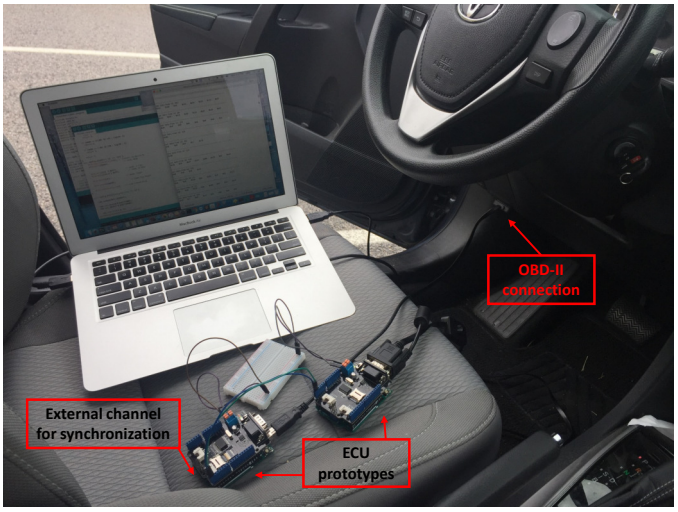


Fig. 7: Prototype connected to a car's CAN bus.

tag at the receiver. We note that although the CAN bus enforces strict message processing deadlines, the individual communication delay encountered by each *message type* (which denotes all messages with the same message identifier) on the bus can vary significantly between 5 ms and 1000 ms [18]. Also, recall that CuMAC supports accumulation of cryptographic strength as per the flexible delay requirement as shown in Figure 4a. This implies that with CuMAC, some message types can be authenticated with high cryptographic strength. Unlike CuMAC, the truncated MAC does not support accumulation of cryptographic strength, and the compound/aggregate MAC cannot provide real-time authentication guarantee which is essential for the CAN bus.

VI. Implementation Results

We discuss the results from the experiments performed with a prototype implementation. These results are also utilized to compare CuMAC with the MAC schemes in the prior art.

A. Details of the Prototype Implementation

Figure 7 illustrates the prototype implementation and the setup that were used for running our experiments. The prototype implementation comprised of two ECU prototypes connected to the on-board diagnostics (OBD) port of the CAN bus (with the bus speed of 500 kbps) of a 2016 Toyota Corolla LE. The ECU prototype consisted of an Arduino UNO board and a Seeed studio CAN shield. The Arduino UNO board was used to emulate the controller unit of an ECU, and the Seeed studio CAN shield worked as the interface between the Arduino UNO board and the CAN bus. The Arduino UNO board utilizes an Atmel ATmega328P chip, which includes a low-power 8-bit micro-controller running at 16 MHz clock speed along with a 32 KB flash memory and a 2 KB RAM. These specifications of the ECU prototype are representative of a typical state-of-the-art automotive-grade controller [34].

With the above experimental setup, we compared five schemes: the trailing MAC, the truncated MAC, the compound MAC, the aggregate MAC and CuMAC. For all five schemes, AES-CMAC with a MAC output of 128 bits was utilized as the underlying MAC algorithm. We utilized an open-source cryptography library [35] to implement AES-CMAC. We found that the computation time (calculated by averaging the computation time over 1000 executions) of generating a MAC was 0.786 ms. For the truncated MAC, the compound MAC, the aggregate MAC and CuMAC, the size of the tag was set to 16 bits, and the message and tag were inserted into the data field of the same CAN packet. For the trailing MAC, the 128-bit MAC was split into two tags of 64 bits, and inserted into the data fields of two consecutive CAN packets. These packets were transmitted immediately after the CAN packet containing only the message.

To evaluate the delay performance, we utilized one ECU prototype (called Tx-ECU) to transmit 6-byte messages with the tags on the CAN bus, and another ECU prototype (called Rx-ECU) to measure the end-to-end delay. In the experiment, the Rx-ECU requested the Tx-ECU (through an external synchronization channel) to send a message, and started the timer. The Rx-ECU stopped the timer after verifying the tag and authenticating the message. The delay was measured as the time between starting the timer and stopping the timer. We let the Rx-ECU trigger the transmission of messages with a period of 10 ms. Also, we let that the message processing deadline for the message type utilized in the experiment be 50 ms. Note that the processing deadline represents the time within which the authentication tags corresponding to the message are expected to be generated, communicated and verified.

B. Results

Table IV summarizes the results from the experiments. The end-to-end delay shown in the table is the worst case delay in processing 1000 CAN packets. The table also presents the cryptographic strengths for real-time, full and partially accumulated authentication in each scheme. From Table IV, we make four important observations: (1) In comparison to the truncated MAC, additional 112 bytes of storage is required to

store the segments of the MACs of seven previous messages in CuMAC. (2) Unlike the trailing MAC, CuMAC does not increase the bus load significantly. (3) Unlike the compound MAC and the aggregate MAC, CuMAC provides real-time authentication with the cryptographic strength of 16 bits. (4) In comparison with the truncated MAC, the compound MAC and the aggregate MAC schemes, CuMAC provides higher cryptographic strength for partially accumulated authentication by enabling the verification of four tags contained in the four packets transmitted (with the periodicity of 10 ms) within the processing deadline of 50 ms.

VII. Conclusion

We proposed a novel concept for message authentication that we refer to as *cumulative MAC* (CuMAC). CuMAC incurs low communication overhead, and provides high cryptographic strength which is commensurate with the delay in authentication. Our promising simulation and experimental results show that CuMAC provides significant advantages over the MAC schemes in the prior art when deployed in a number of emerging applications, including those that run on energy-constrained or bandwidth-constrained networks.

Acknowledgement

This work was partially sponsored by National Science Foundation (NSF) through grants 1563832, 1642928, and 1822173, and by the industry affiliates of the Broadband Wireless Access & Applications Center (BWAC).

References

[1] U. Raza, P. Kulkarni, and M. Sooriyabandara, "Low power wide area networks: An overview," *IEEE Communications Surveys Tutorials*, vol. 19, no. 2, pp. 855–873, 2017.

[2] R. Escherich, I. Ledendecker, C. Schmal, B. Kuhls, C. Grothe, and F. Scharberth, "SHE: Secure hardware extension - Functional specification, Version 1.1," *Hersteller Initiative Software (HIS) AK Security*, 2009.

[3] R. Soja. Automotive security: From standards to implementation. Accessed: July 1, 2019. [Online]. Available: https://www.nxp.com/docs/en/white-paper/AUTOSECURITYWP.pdf

[4] D. K. Nilsson, U. E. Larson, and E. Jonsson, "Efficient in-vehicle delayed data authentication based on compound message authentication codes," in *IEEE 68th Vehicular Technology Conference*, 2008, pp. 1–5.

[5] H. Wang and A. O. Fapojuwo, "A survey of enabling technologies of low power and long range machine-to-machine communications," *IEEE Communications Surveys Tutorials*, vol. 19, no. 4, pp. 2621–2639, 2017.

[6] C. Szilagyi and P. Koopman, "A flexible approach to embedded network multicast authentication," in *Proceedings of the 2nd Workshop on Embedded Systems Security (WESS)*, 2008.

[7] H. Schweppe, Y. Roudier, B. Weyl, L. Apvrille, and D. Scheuermann, "Car2x communication: securing the last meter-a cost-effective approach for ensuring trust in car2x applications using in-vehicle symmetric cryptography," in *2011 IEEE Vehicular Technology Conference (VTC Fall)*. IEEE, 2011, pp. 1–5.

[8] K. Bhargavan and G. Leurent, "Transcript collision attacks: Breaking authentication in TLS, IKE, and SSH," in *Network and Distributed System Security Symposium (NDSS)*, 2016.

[9] J. Katz and A. Lindell, "Aggregate message authentication codes," *Topics in Cryptology–CT-RSA*, pp. 155–169, 2008.

[10] C. Gomez, J. Oller, and J. Paradells, "Overview and evaluation of Bluetooth Low Energy: An emerging low-power wireless technology," *Sensors*, vol. 12, no. 9, pp. 11 734–11 753, 2012.

[11] C. Bormann, A. P. Castellani, and Z. Shelby, "CoAP: An application protocol for billions of tiny internet nodes," *IEEE Internet Computing*, no. 2, pp. 62–67, 2012.

[12] M. Singh, M. Rajan, V. Shivraj, and P. Balamuralidhar, "Secure MQTT for Internet of Things (IoT)," in *Fifth International Conference on Communication Systems and Network Technologies*, 2015, pp. 746–751.

[13] Sigfox. Technical overview. Accessed: July 1, 2019. [Online]. Available: https://www.disk91.com/wp-content/uploads/2017/05/4967675830228422064.pdf

[14] R. Roman, J. Zhou, and J. Lopez, "On the features and challenges of security and privacy in distributed internet of things," *Computer Networks*, vol. 57, no. 10, pp. 2266–2279, 2013.

[15] R. Bosch, "CAN specification - Version 2.0," 1991.

[16] International Organization for Standardization, "ISO/IEC 11898-1:2015: Road vehicles - Controller area network (CAN) - Part 1: Data link layer and physical signalling," Standard.

[17] K. H. Johansson, M. Törngren, and L. Nielsen, "Vehicle applications of controller area network," in *Handbook of Networked and Embedded Control Systems*, 2005, pp. 741–765.

[18] G. M. Zago and E. P. de Freitas, "A quantitative performance study on CAN and CAN FD vehicular networks," *IEEE Transactions on Industrial Electronics*, vol. 65, no. 5, pp. 4413–4422, 2018.

[19] S. Checkoway, D. Mccoy, B. Kantor, D. Anderson, H. Shacham, S. Savage, K. Koscher, A. Czeskis, F. Roesner, and T. Kohno, "Comprehensive experimental analyses of automotive attack surfaces," in *Proceedings of the 20th USENIX Security Symposium*, 2011.

[20] K. Koscher, A. Czeskis, F. Roesner, S. Patel, T. Kohno, S. Checkoway, D. McCoy, B. Kantor, D. Anderson, H. Shacham, and S. Savage, "Experimental security analysis of a modern automobile," in *IEEE Symposium on Security and Privacy*, 2010, pp. 447–462.

[21] C. Miller and C. Valasek, "Remote exploitation of an unaltered passenger vehicle," *Black Hat USA*, 2015.

[22] National Highway Traffic Safety Administration, "Cybersecurity best practices for modern vehicles," *No. DOT HS 812*, vol. 333, 2016.

[23] H. Ueda, R. Kurachi, H. Takada, T. Mizutani, M. Inoue, and S. Horihata, "Security authentication system for in-vehicle network," *SEI Technical Review*, no. 81, 2015.

[24] F. Wang and J. Liu, "Networked wireless sensor data collection: Issues, challenges, and approaches," *IEEE Communications Surveys Tutorials*, vol. 13, no. 4, pp. 673–687, 2011.

[25] D. J. Bernstein and T. Lange, "Non-uniform cracks in the concrete: the power of free precomputation," in *International Conference on the Theory and Application of Cryptology and Information Security*, 2013, pp. 321–340.

[26] H. Li, V. Kumar, J.-M. Park, and Y. Yang, "Cumulative message authentication codes for resource-constrained networks," *arXiv preprint arXiv:2001.05211*, 2020.

[27] W. Du, J. Deng, Y. S. Han, P. K. Varshney, J. Katz, and A. Khalili, "A pairwise key predistribution scheme for wireless sensor networks," *ACM Transactions on Information and System Security (TISSEC)*, vol. 8, no. 2, pp. 228–258, 2005.

[28] M. Bellare, J. Kilian, and P. Rogaway, "The security of the cipher block chaining message authentication code," *Journal of Computer and System Sciences*, vol. 61, no. 3, pp. 362–399, 2000.

[29] R. A. Rohde and R. A. Muller, "Air pollution in China: Mapping of concentrations and sources," *PloS One*, vol. 10, no. 8, pp. 1–14, 2015.

[30] ON Semiconductor. Ultra-low power, AT command controlled, Sigfox compliant transceiver IC for up-link and down-link. Accessed: July 1, 2019. [Online]. Available: https://www.onsemi.com/pub/Collateral/AX-SIGFOX-D.PDF

[31] S. Kramer, D. Ziegenbein, and A. Hamann, "Real world automotive benchmarks for free," in *6th International Workshop on Analysis Tools and Methodologies for Embedded and Real-time Systems (WATERS)*, 2015.

[32] B. Groza, S. Murvay, A. V. Herrewege, and I. Verbauwhede, "LiBrA-CAN: A lightweight broadcast authentication protocol for controller area networks," in *Cryptology and Network Security*, 2012, pp. 185–200.

[33] R. I. Davis, A. Burns, R. J. Bril, and J. J. Lukkien, "Controller area network (CAN) schedulability analysis: Refuted, revisited and revised," *Real-Time Systems*, vol. 35, no. 3, pp. 239–272, 2007.

[34] P.-S. Murvay, A. Matei, C. Solomon, and B. Groza, "Development of an AUTOSAR compliant cryptographic library on state-of-the-art automotive grade controllers," in *11th IEEE International Conference on Availability, Reliability and Security (ARES)*, 2016, pp. 117–126.

[35] Arduino cryptography library. Accessed: July 1, 2019. [Online]. Available: https://github.com/rweather/arduinolibs