Cumulative Message Authentication Codes for Resource-Constrained IoT Networks

He Li, Vireshwar Kumar, *Member, IEEE,* Jung-Min (Jerry) Park, *Fellow, IEEE,* and Yaling Yang, *Member, IEEE*

***Abstract*—In resource-constrained IoT networks, 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 of the underlying MAC by collecting and verifying the authentication tags. Embodied with these two procedures, CuMAC enables the receiver to achieve an advantageous trade-off between the cryptographic strength and the latency in the processing of the authentication tags. Furthermore, for some latency-sensitive messages where this trade-off may be unacceptable, we propose a variant of CuMAC that we refer to as *CuMAC with Speculation* (CuMAC/S). In addition to the aggregation and accumulation procedures, CuMAC/S enables the sender and receiver to employ a speculation procedure for predicting future message values and precomputing the corresponding MAC segments. For the messages which can be reliably speculated, CuMAC/S significantly reduces the MAC verification latency without compromising the cryptographic strength. We have carried out a comprehensive evaluation of CuMAC and CuMAC/S through simulation and a prototype implementation on a real car.**

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

I. Introduction

In emerging applications, such as intelligent automobiles, industrial control systems, and smart city 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/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.

Problem. 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 and quality of the key, and the size of the MAC output. Hence, a *conventional MAC* scheme typically employs at least a few hundred bits of MAC output to ensure a sufficient level of cryptographic strength. Unfortunately, in resource-constrained IoT networks (e.g., energy-constrained low-power wide-area network with battery-powered devices and bandwidth-constrained in-vehicle controller area network), the payload size of each packet is very short, i.e., less than a hundred bits [4]. As such, not more than a few bits can be spared to include an *authentication tag*, prohibiting the usage of the conventional MAC [1].

Related Work. 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, the truncated MAC sacrifices cryptographic strength in exchange for reduced communication overhead and energy consumption, 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 validity of the MAC.

Challenges. In the above discussion, we identify three critical challenges in employing MACs for IoT networks: (1) incurring minimal communication overhead so that the MAC can fit in a packet, (2) ensuring that the cryptographic strength meets the security need of the application, and (3) incurring minimal latency so that the MAC generation and verification processes do not cause unacceptable delays in the packet processing.

Proposed Solution. In this paper, we address the afore-

H. Li, J. Park, and Y. Yang are with the Department of Electrical and Computer Engineering, Virginia Tech, Blacksburg, Virginia, USA (e-mail: heli@vt.edu, jungmin@vt.edu, yyang8@vt.edu).

V. Kumar is with the Department of Computer Science and Engineering, Indian Institute of Technology, Delhi, India (e-mail: viresh@cse.iitd.ac.in).

mentioned challenges through a novel approach for message authentication that we refer to as *Cumulative Message Authentication Code* (CuMAC). In CuMAC, a sender utilizes a procedure called *aggregation* through which the sender first divides the full-sized MAC 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 a 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 and the third challenge, it brings up a novel and flexible trade-off between the cryptographic strength and latency. CuMAC enables the receiver to authenticate the message in real-time with the cryptographic strength which is commensurate with the size of each tag. Meanwhile, CuMAC also enables the authentication with the highest level of cryptographic strength after accumulating all segments of the MAC that cover the message in the associated packets.

Moreover, in latency-sensitive IoT applications, the receiver may be required to immediately authenticate a message with high cryptographic strength as it arrives. In such cases, the trade-off made by CuMAC may not be sufficient. To address this need, we propose a variant of CuMAC called *CuMAC with Speculation* (*CuMAC/S*) that enables a receiver to accumulate the MAC's cryptographic strength while incurring a minimal delay. The core concept of CuMAC/S is motivated by the technique of *speculative execution*[1] which is widely employed in modern computer systems [10], [11]. CuMAC/S can be utilized in IoT applications where future messages can be predicted correctly with high reliability with an appropriate speculation model using the current and past messages.

In CuMAC/S, a sender speculates future messages, computes the corresponding MACs, and aggregates the MAC segments of the speculated messages into the authentication tag of the current packet. If the speculated value of a received message is equal to the actual value, then all its segments can be verified in current and previous tags, and hence the receiver can *accumulate* cryptographic strength without having to wait for tags included in forthcoming packets; this significantly cuts down on the MAC verification delay.

The paper's main contributions are summarized as follows.

- We propose a novel message authentication scheme called *CuMAC*, which meets the security need of resource-constrained IoT applications. CuMAC is an embodiment of two novel concepts that we refer to as *aggregation* (which reduces the communication overhead) and *accumulation* (which increases the cryptographic strength).
- We propose a variant of CuMAC called CuMAC/S that meets the security need of delay-sensitive, resource-constrained IoT applications. CuMAC/S enables the accumulation of cryptographic strength while incurring minimal delay by employing the novel idea of speculation.
- We have thoroughly evaluated the effectiveness of CuMAC and CuMAC/S through a simulated in-vehicle controller area network and a prototype implementation on a real car. 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 increased latency. Further, for the messages which can be accurately speculated, CuMAC/S achieves the cryptographic strength equivalent to the conventional MAC scheme without any additional latency.

[1]Speculative execution is an optimization technique in which a computer system performs speculative execution where some outcome is predicted and execution proceeds along a predicted path. Work is done before it is known whether it is actually needed, so as to prevent a delay that would have to be incurred by doing the work after it is known that it is needed.

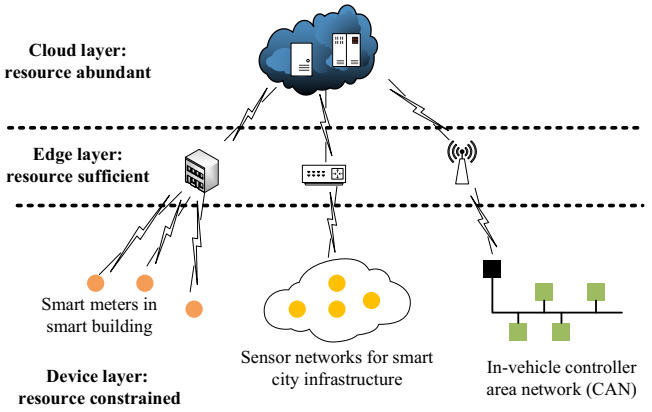


Fig. 1: Architecture of typical IoT networks.

II. Motivation for Short MACs

IoT networks consist of resource-constrained devices at the lowest layer as shown in Figure 1. To enable message authentication in such networks, it is imperative to use short MACs as demonstrated by the following discussion of two specific application scenarios – one with the energy-constrained devices and another with the bandwidth-constrained devices.

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

Many IoT applications (e.g., smart metering and smart city infrastructure) require a densely deployed 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 [12] is one example of a widely-known LPWAN technology. In Sigfox, each uplink packet contains a counter, a message (with the length between 0 and 96 bits), and an authentication tag (with the 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, the sender waits for an acknowledgment from the receiver. In the

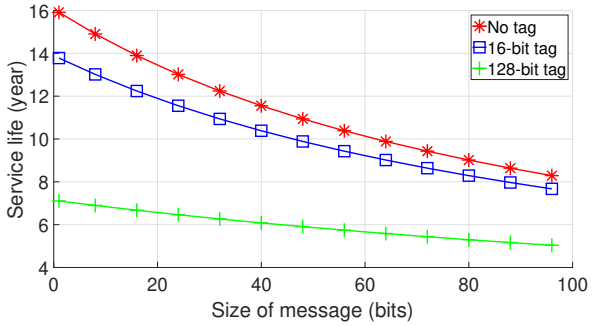


Fig. 2: Effect of the size of message and authentication tag on the service life of a sensor node in Sigfox (the results are obtained using the battery consumption data from a Sigfox compliant transceiver produced by ON Semiconductor [13].)

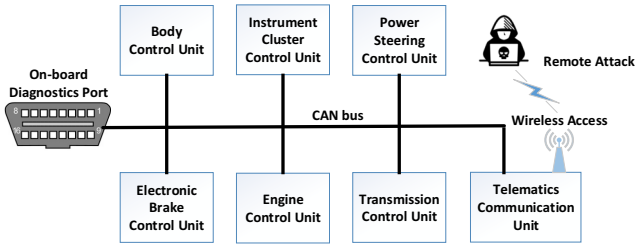


Fig. 3: Architecture of an in-vehicle controller area network.

Fig. 4: Considered packet model.

absence of the acknowledgment, the packet is considered lost. Sigfox does not support retransmission of such 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 communicated packets, it is imperative to communicate using short packets to ensure long service life. Figure 2 illustrates that in comparison to the standard benchmark of 48-bit messages without any tags, while utilizing a short MAC with 16-bit tags achieves a modest (around 10%) reduction in the service life, utilizing the conventional MAC with 128-bit tags results in a significant loss of around 45% of the service life. As such, although the message integrity and authentication are of prime importance in applications supported by Sigfox [14], the energy overhead of communicating the full-sized MAC output in the Sigfox packet is undesirably high.

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 adaptive cruise control and internet-of-vehicles (IoV) [15]–[17]. As shown in Figure 3, these ECUs communicate with each other over a bandwidth-constrained wired broadcast channel called the CAN bus [18], [19]. Because the messages communicated among ECUs directly affect vital functions of a vehicle, some of which are safety-related (e.g., dynamics control system [20]), 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 acknowledgment and retransmission of corrupted/lost packets, it does not support any security mechanism [21].

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 [22], [23]. Due to one such vulnerability, Jeep had to recall 1.4 million vehicles in 2015 [24]. Although the existing literature [25]–[27] addresses some of the authentication issues in a vehicle-to-smart grid (V2G) or internet-of-vehicles (IoV) scenarios, there is a lack of an effective and practical scheme to counter impersonation attacks against ECUs in CAN. To this end, the US National Highway Traffic Safety Administration (NHTSA) recommends the inclusion of MACs for the end-to-end authentication of the messages on the CAN bus [28].

A CAN packet consists of an 11-bit or a 29-bit identifier field and a message field with a length between 0 and 64 bits. Except for 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], [29], 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. Although such a design of the 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. Further, since the bandwidth of a typical CAN bus is only 500 kbit/s, transmitting additional trailing packets containing only the authentication tag undesirably increases the bus load [30].

Proposed Design. In both the above application scenarios (LPWAN and CAN), the constraints of the IoT network – either in terms of the MAC size or energy/bandwidth consumption of the networked devices – prohibit the use of the conventional MAC scheme. To address this challenge, we propose CuMAC and CuMAC/S which can be readily employed in these scenarios to achieve the desired level of security provided by short MACs. In this paper, we utilize CAN as a concrete application scenario to highlight the advantages of CuMAC and CuMAC/S. However, these two schemes can be applied to other resource-constrained network applications, including IoT applications (e.g. LPWAN, Bluetooth Low Energy (BLE) [31], Constrained Access Protocol (CoAP) [32] or Message Queue Telemetry Transport (MQTT) [33]).

III. Model and Security Objectives

Here we discuss the network model and define the security objectives of the proposed MAC schemes.

A. Model and Assumptions

System Model. We consider an energy-constrained and/or bandwidth-constrained IoT network where a sender needs to transmit security-critical messages to a receiver using small packets. Hence, the sender and the receiver (after sharing a secret key) employ a MAC scheme for message authentication. As shown in Figure 4, we let the sender employ a packet format that 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 and CuMAC/S. If the network protocol (e.g., Sigfox as discussed in Section II-A) employs these fields in the conventional packets by design, we 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 acknowledgment mechanism that enables the sender to know if a particular packet was correctly delivered to the receiver [34]. The acknowledgment 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.

Threat Model. We consider an adversary that aims to forge valid authentication tags for its malicious messages so that it can deceive the authentication scheme at the receiver. While the adversary can eavesdrop on the communication channel to obtain packets transmitted by the sender, it does not know the secret key (used for generating and verifying authentication tags) shared between the sender and the receiver.

Cryptographic Strength. 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 against the scheme is negligibly small [35]. The cryptographic strength of a conventional MAC depends on three security criteria: (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. In this paper, we assume that the first and second criteria have been satisfied, and focus only on the third criterion. As such, to achieve a cryptographic strength of λ bits, the minimum size of the MAC output (denoted by L) should be λ bits.

B. Proposed Approach and Security Objectives

MAC Design. As shown in Figure 5, we consider that an L-bit MAC of a message m_i is divided into n segments each of length l, and distributed in tags $\tau_i, \cdots, \tau_{i+n-1}$. Also, if after generating the message m_{i-n+1}, the message m_i can be speculated as $\widehat{m}_i$, the corresponding MAC is computed as $\widehat{\sigma}_i$. The MAC $\widehat{\sigma}_i$ is divided into n segments, and the last $n-1$ segments are distributed in tags $\tau_{i-n+1}, \cdots, \tau_{i-1}$, which are transmitted before τ_i.

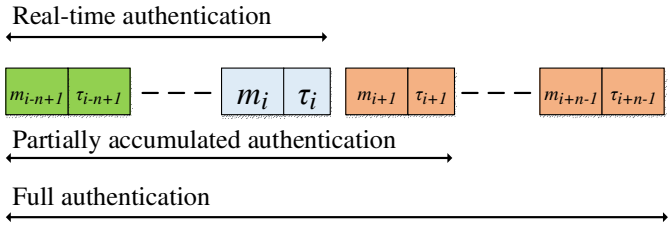


Fig. 5: Illustrative distribution of the segments of the MAC of message m_i, and definition of the authentication levels.

Authentication Levels. To compare the proposed approach with the prior art, we define three levels/features of authentication: (1) real-time authentication, (2) full authentication, and (3) partially accumulated authentication. Figure 5 illustrates these different levels of authentication, when applied to message m_i. Here the receiver can perform real-time authentication immediately after receiving message m_i by processing the current tag τ_i and the previous tags $\tau_{i-n+1}, \cdots, \tau_{i-1}$. 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 $\tau_{i-n+1}, \cdots, \tau_{i+n-1}$. 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 $\tau_{i-n+1}, \cdots, \tau_{i+r-1}$, 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.

Security Objectives. The security objective of the proposed MAC scheme is to ensure that the probability with which an adversary succeeds in breaking each of the three authentication features is negligible (i.e, as difficult as random guessing). Specifically, to break the real-time authentication feature, the adversary needs to forge a message and a valid tag. The forgery needs to be *fresh* which means that the sender has not generated the MAC of the same counter and message pair using the same shared key. As such, the cryptographic strength of real-time authentication depends on the size of the MAC segment l. To break the partially accumulated authentication feature with r accumulated segments, the adversary needs to forge a sequence of r messages with valid tags. In this sequence, the forgery for only the first message needs to be fresh. Hence, the cryptographic strength of partially accumulated authentication depends on the size of the MAC segment l and the number of accumulated segments r. Similarly, to break the full authentication feature, the adversary needs to forge a sequence of n messages with valid tags, where forgery for at least the first message is fresh. Hence, the cryptographic strength of full authentication is limited by the size of MAC L. In the case of the speculation of future messages, the cryptographic strengths of the real-time and the partially accumulated authentication also depend on the message speculation accuracy.

A formal discussion of the security properties and associated

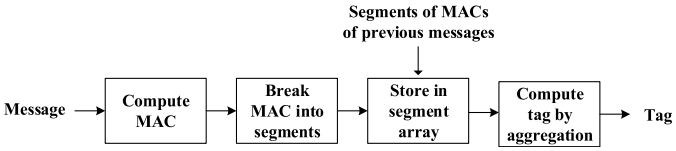


Fig. 6: Schematic of the procedures at the sender in CuMAC.

proofs corresponding to CuMAC and CuMAC/S are provided in Appendix A and B.

IV. Technical Details of CuMAC

CuMAC comprises two major algorithms: tag generation and tag verification. In the tag generation algorithm, the sender computes the authentication tag through two major steps (Figure 6). In the first step, the sender generates the MAC of the message, breaks the MAC into short segments, and stores them into a segment array. In the second step, the sender retrieves one MAC segment of the current message, and several segments of the MACs of the previously transmitted messages from the segment array, and aggregates the segments to generate a tag. 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 segment (aggregated in the authentication tag) with the previously received MAC segments of the corresponding message.

Below we present the technical details of the algorithms in CuMAC, and an instantiation that illustrates the generation and verification of the tags in CuMAC.

A. Algorithms

CuMAC is composed of the following algorithms, where the $\mathsf{KeyGen}(1^\lambda)$ and $\mathsf{MacGen}(\mathtt{k}, i, m_i)$ algorithms utilize existing approaches and the rest ones are proposed in this paper.

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

This probabilistic key generation algorithm is utilized by the sender and 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 node [7]. In the absence of such a node, it can also be realized using an efficient key distribution scheme [36].

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

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 [37]) or a hash-based (e.g., SHA-3) MAC scheme. In this paper, we utilize the widely used AES-CMAC.

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

This segment aggregation algorithm is utilized by the sender and the receiver as a sub-algorithm of tag generation and tag verification algorithms, respectively. It takes as input a two-dimensional array of MAC segments $\mathtt{segArray}$. This algorithm proceeds as follows. 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

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

It means 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.

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

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 $\mathtt{segTx}$.
3) Compute and output the tag τ_i by aggregating the segments of MACs in $\mathtt{segTx}$ as shown in equation (2), i.e., $\tau_i \leftarrow \mathsf{SegAgg}(\mathtt{segTx})$.

After receiving a positive acknowledgment of the delivery of the packet at the receiver, the sender increments the counter i by one for the next packet. We note that the 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 acknowledgment from the receiver or it receives a negative acknowledgment. In this case, 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 $\mathtt{segTx}$, and then proceeds with the tag generation of the next message.

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

This verification algorithm is run by the receiver for verifying the authenticity of the received message and tag by regenerating the authentication tag and compared to the received tag. It takes as inputs the secret key $\mathtt{k}$, the received counter i, the received message m_i, and the received tag τ_i. It also utilizes an array of MAC segments $\mathtt{segRx}$ and an array of verified MAC segments $\mathtt{accRx}$. These arrays are stored and maintained by the receiver. This algorithm first generates the tag for the received message using the TagGen algorithm while updating

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

the MAC segments in segRx, i.e., $\widetilde{\tau}_i \leftarrow \mathsf{TagGen}(\mathtt{k}, i, m_i)$. It then verifies whether the generated tag $\widetilde{\tau}_i$ is equal to the received tag τ_i. If the verification succeeds, it updates the array of accumulated MAC segments accRx and outputs the value valid; otherwise, it outputs the value invalid.

B. Illustration

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., $n = 4$). 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_3. 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.

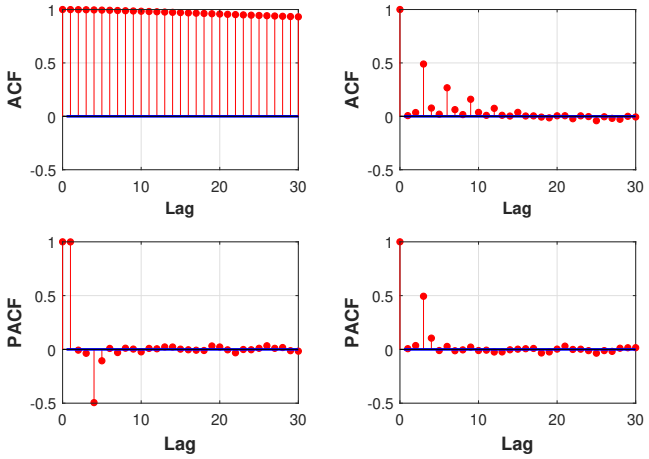
When the receiver receives the fifth packet with the message m_5, the successful verification of the tag τ_5 enables the real-time authentication of message m_5 with the cryptographic strength of 32 bits. Next, the receiver receives and verifies the validity of tags τ_6, τ_7, and τ_8. If all four tags 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. We highlight that this ability to perform the partially accumulated authentication is the most unique feature of CuMAC when compared to the prior art.

V. Technical Details of CuMAC/S

In latency-sensitive applications, the receiver must authenticate a message as it arrives. For such applications, the trade-off between the cryptographic strength and latency made by CuMAC may not be sufficient. To address this challenge, we present CuMAC/S, which employs a novel concept of message speculation for MAC generation. Equipped with an accurate message speculation algorithm, CuMAC/S achieves both high cryptographic strength and low verification latency.

A. Feasibility of Speculation

We discuss the feasibility of speculation of future messages by analyzing the messages communicated on a CAN bus in a typical vehicle. To evaluate the speculation accuracy for different CAN messages, we utilize trace files of a real vehicle, which have been recorded using the OpenXC platform [38]. These files present different types of CAN messages which can be identified and interpreted by OpenXC libraries. To speculate future message values, we utilize the autoregressive integrated moving average (ARIMA) model, which is a widely used model for time series analysis.



(a) Original data. (b) First-order differenced data.

Fig. 7: Example illustrating the feasibility of speculation of a vehicle's transmission torque values through an ARIMA model whose parameters are determined using the autocorrelation function (ACF) and partial autocorrelation function (PACF).

The ARIMA model with hyperparameters (p, d, q) implies that for the d^{th}-order difference of the time series values, a speculated future message value is the linear combination of p previous values, and q previous error values in the speculation. We utilize the Box-Jenkins [39] method to compute the hyperparameters of the ARIMA model for each type of CAN message. In this method, we determine the values for p, d, and q by observing the autocorrelation and partial autocorrelation of the message values. We illustrate this procedure in Figure 7 which presents the autocorrelation and partial autocorrelation of the values of the message corresponding to the torque at transmission in a vehicle. From the results shown in Figure 7a, we observe that there is high autocorrelation between message values. Further, from the results shown in Figure 7b, we observe that the autocorrelation decays gradually, and the partial autocorrelation is close to zero after a lag of 3 message values. Hence, according to the rules of the Box-Jenkins approach, we set ARIMA(3,1,0) model to speculate the message values corresponding to the torque at the transmission.

In our analysis, we train the ARIMA model using the first 90% of the message values, and then we employ the model on the last 10% of message values for the test. Here, the accuracy of correct speculation/prediction of future message values is measured using a metric called speculation error rate (SER), which is defined as the ratio between the number of incorrect speculations and the total number of speculations. Note that the speculation is correct only if the message is correctly

TABLE II: Speculation accuracy for typical CAN messages.

Signal	SER	SER after ignoring 3 LSBs
Longitude	<0.0001	<0.0001
Latitude	<0.0001	<0.0001
Odometer	<0.0001	<0.0001
Fuel level	<0.0001	<0.0001
Fuel consumed since restart	<0.0001	<0.0001
Accelerator pedal position	0.0030	0.0002
Torque at transmission	0.0100	0.0020
Engine speed	0.2329	0.0880
Vehicle speed	0.2478	0.0975
Steering wheel angle	0.4763	0.3595

predicted up to the least significant bit (LSB). Table II shows the speculation error rate of ten message-types. We observe that certain types of CAN messages (e.g. the first five message types listed in Table II) can be predicted with high reliability using the ARIMA model.

We can improve the speculation accuracy by using more sophisticated and further tuned models. Moreover, we can mitigate the impact of speculation errors by increasing the robustness of the MAC scheme against such errors. For example, if the message contains some values for which some of the least significant bits can be safely ignored (without impacting performance or security), then these bits do not need to be protected by a MAC, and hence the MAC calculation can be limited to only the part of a message that can be predicted with high reliability. In the rightmost column of Table II, we show that the SER can be significantly improved for some types of messages by ignoring the last three least LSBs.

Now we present the technical details of the algorithms in CuMAC/S and an instantiation that illustrates the generation and verification of the tags in CuMAC/S.

B. Algorithms

The KeyGen and MacGen algorithms in CuMAC (discussed in Section IV) and those in CuMAC/S are the same, and hence we do not provide their details in this section. We present the details of other algorithms in CuMAC/S as follows.

$\texttt{msgArray}' \leftarrow \textbf{MsgSpec}(\texttt{msgArray})$

This deterministic message speculation algorithm is utilized by the sender and receiver for the speculation of future message values. It takes an array of the transmitted and speculated messages `msgArray` as input. At the i^{th} instance, the array `msgArray` can be represented as $\{m_1, m_2, \cdots, m_{i-1}, m_i, \widehat{m}_{i+1}, \widehat{m}_{i+2}, \cdots \widehat{m}_{i+n-2}\}$. This algorithm generates the predicted value of the message m_{i+n-1}, which is represented by $\widehat{m}_{i+n-1}$, appends it to the array `msgArray`, and outputs the updated array $\texttt{msgArray}'$. Since the speculation model used in this algorithm is deterministic, the sender and the receiver run the same set of steps, and obtain the same speculated messages given the same input messages.

$\tau_i \leftarrow \textbf{SegAgg}(\texttt{segArray})$

This segment aggregation algorithm is run by the sender and the receiver. It takes as input a two-dimensional array of MAC segments `segArray`. The `segArray` comprises of the segments of the MACs of the transmitted and speculated messages. The i^{th} entry in `segArray` is generated by the segment $s_i^j \ \forall j \in [1, n]$ using the equation (1). This algorithm extracts $2n-1$ elements from `segArray` ($n-1$ previous MAC segments, current MAC segment, and $n-1$ speculated MAC segments), and computes the authentication tag τ_i as follows.

$$\tau_i \leftarrow \left(\bigoplus_{j=1, i-j+1>0}^{n} s_{i-j+1}^{j} \right) \oplus \left(\bigoplus_{j=2}^{n} \widehat{s}_{i+j-1}^{j} \right). \quad (3)$$

This algorithm outputs the authentication tag τ_i.

$\tau_i \leftarrow \textbf{TagGen}(\texttt{k}, i, m_i)$

This tag generation algorithm is utilized by the sender to generate an authentication tag. It takes as inputs the secret key `k`, a counter i and a message m_i. It utilizes an array of the transmitted and speculated messages `msgTx`, and an array of MAC segments of the transmitted and speculated messages `segTx`. The arrays `msgTx` and `segTx` are stored and maintained by the sender. Figure 8 presents an overview of the algorithm which proceeds as follows.

1) Extract $\widehat{m}_i$ from `msgTx` and verify whether $m_i = \widehat{m}_i$.
   a) If $m_i = \widehat{m}_i$, set $\sigma_i = \widehat{\sigma}_i$.
   b) Otherwise, if $m_i \neq \widehat{m}_i$, compute the MAC of the message m_i and set it as σ_i, i.e., $\sigma_i \leftarrow \mathsf{MacGen}(\texttt{k}, i, m_i)$. Divide the MAC σ_i into n segments and replace the MAC segments of $\widehat{\sigma}_i$ in the array `segTx`.
2) Predict the value of the message m_{i+n-1} and append the speculated message $\widehat{m}_{i+n-1}$ to the array `msgTx`, i.e., $\texttt{msgTx}' \leftarrow \mathsf{MsgSpec}(\texttt{msgTx})$.
3) Compute the MAC of the message $\widehat{m}_{i+n-1}$ and set it as $\widehat{\sigma}_{i+n-1}$, i.e., $\widehat{\sigma}_{i+n-1} \leftarrow \mathsf{MacGen}(\texttt{k}, i, \widehat{m}_{i+n-1})$. Divide the MAC $\widehat{\sigma}_{i+n-1}$ into n segments and append to the array `segTx`.
4) Compute the current tag τ_i by aggregating the segments of MACs of the previous, current and future messages, i.e., $\tau_i \leftarrow \mathsf{SegAgg}(\texttt{segTx})$.

$\texttt{valid/invalid} \leftarrow \textbf{TagVerify}(\texttt{k}, i, m_i, \tau_i)$

This verification algorithm is run by the receiver for verifying the authenticity of the received message and tag by regenerating the authentication tag and compared to the received 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 stores and manages an array of previously received and speculated messages `msgRx`, an array of MAC segments `segRx`, and an array of verified segments `accRx`. This algorithm first generates the tag for the received message using the TagGen

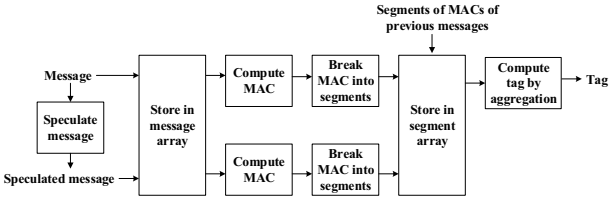


Fig. 8: Schematic of the procedures at the sender in CuMAC/S.

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

Packet Counter	Previous MACs	Current MAC	Previous speculated MACs	Current speculated MAC	Aggregation of MAC segments	Tag
2	σ_1	σ_2	$\widehat{\sigma}_3, \widehat{\sigma}_4$	$\widehat{\sigma}_5$	$s_1^2 \oplus s_2^1 \oplus \widehat{s}_3^2 \oplus \widehat{s}_4^3 \oplus \widehat{s}_5^4$	τ_2
3	σ_1, σ_2	σ_3	$\widehat{\sigma}_4, \widehat{\sigma}_5$	$\widehat{\sigma}_6$	$s_1^3 \oplus s_2^2 \oplus s_3^1 \oplus \widehat{s}_4^2 \oplus \widehat{s}_5^3 \oplus \widehat{s}_6^4$	τ_3
4	$\sigma_1, \sigma_2, \sigma_3$	σ_4	$\widehat{\sigma}_5, \widehat{\sigma}_6$	$\widehat{\sigma}_7$	$s_1^4 \oplus s_2^3 \oplus s_3^2 \oplus s_4^1 \oplus \widehat{s}_5^2 \oplus \widehat{s}_6^3 \oplus \widehat{s}_7^4$	τ_4
5	$\sigma_2, \sigma_3, \sigma_4$	σ_5	$\widehat{\sigma}_6, \widehat{\sigma}_7$	$\widehat{\sigma}_8$	$s_2^4 \oplus s_3^3 \oplus s_4^2 \oplus s_5^1 \oplus \widehat{s}_6^2 \oplus \widehat{s}_7^3 \oplus \widehat{s}_8^4$	τ_5
6	$\sigma_3, \sigma_4, \sigma_5$	σ_6	$\widehat{\sigma}_7, \widehat{\sigma}_8$	$\widehat{\sigma}_9$	$s_3^4 \oplus s_4^3 \oplus s_5^2 \oplus s_6^1 \oplus \widehat{s}_7^2 \oplus \widehat{s}_8^3 \oplus \widehat{s}_9^4$	τ_6
7	$\sigma_4, \sigma_5, \sigma_6$	σ_7	$\widehat{\sigma}_8, \widehat{\sigma}_9$	$\widehat{\sigma}_{10}$	$s_4^4 \oplus s_5^3 \oplus s_6^2 \oplus s_7^1 \oplus \widehat{s}_8^2 \oplus \widehat{s}_9^3 \oplus \widehat{s}_{10}^4$	τ_7
8	$\sigma_5, \sigma_6, \sigma_7$	σ_8	$\widehat{\sigma}_9, \widehat{\sigma}_{10}$	$\widehat{\sigma}_{11}$	$s_5^4 \oplus s_6^3 \oplus s_7^2 \oplus s_8^1 \oplus \widehat{s}_9^2 \oplus \widehat{s}_{10}^3 \oplus \widehat{s}_{11}^4$	τ_8

algorithm while updating the speculated message in `msgRx` and MAC segments in `segRx`, i.e., $\widetilde{\tau}_i \leftarrow \mathsf{TagGen}(\mathtt{k}, i, m_i)$. It then verifies whether the generated tag $\widetilde{\tau}_i$ is equal to the received tag τ_i. If the verification succeeds, it updates the array of accumulated MAC segments `accRx` and outputs the value `valid`; otherwise, it outputs the value `invalid`.

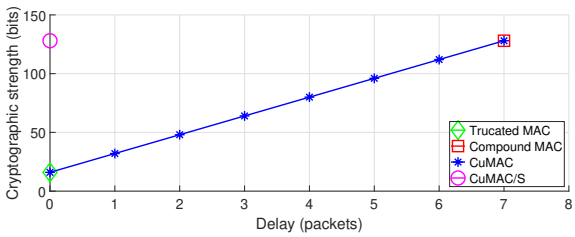
C. Illustration

Table III presents an example of CuMAC/S, which follows the example of CuMAC presented in Section IV-B. When the receiver receives the fifth packet with the message m_5, it verifies whether it matches the speculated message $\widehat{m}_5$. If they do not match, the successful verification of the tag τ_5 enables the real-time authentication of message m_5 with a cryptographic strength of 32 bits, which is the same as in CuMAC. Next, the receiver receives and verifies the validity of tags $\tau_5, \cdots, \tau_8$. 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 a cryptographic strength of 128 $(= 4 \times 32)$ bits.

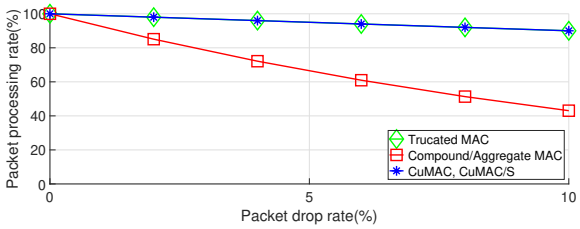
However, if the message m_5 and $\widehat{m}_5$ match, and tags τ_2, τ_3, τ_4 and τ_5 are verified as `valid`, the receiver combines the segments s_5^1, $\widehat{s}_5^2$, $\widehat{s}_5^3$ and $\widehat{s}_5^4$—which are contained in tags τ_5, τ_4, τ_3 and τ_2, respectively. This enables the receiver to perform real-time authentication of message m_5 with a cryptographic strength of 128 bits. We highlight that this unique ability to achieve equal cryptographic strengths for the real-time authentication and full authentication in spite of using short authentication tags distinguishes CuMAC/S from prior art.

VI. Simulation Results

In this section, we consider a simulated IoT environment, where we assume AES-CMAC with a MAC output of 128 bits as the underlying MAC algorithm, and we set the size of the tag in all schemes to 16 bits. We evaluate the performance of CuMAC and CuMAC/S by comparing them with three other schemes from the prior art: the truncated MAC [6], the compound MAC [4], and the aggregate MAC [9]. 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



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



(b) Effect of unreliable communication channel.

Fig. 9: Illustration of the higher cryptographic strength and higher packet processing rate achieved by CuMAC and CuMAC/S in comparison with the prior art.

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 and CuMAC/S, each MAC of 128 bits is divided into eight segments each of size 16 bits. In CuMAC, each tag is generated by aggregating segments of seven previously transmitted messages and the current message. In CuMAC/S, each tag is generated by aggregating segments of seven previously transmitted messages, the current message, and seven speculated messages.

Cryptographic Strength. Figure 9a presents the cryptographic strengths of the MAC 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 for the truncated MAC. As more packets are received, partially accumulated authentication is achieved and CuMAC provides increasing cryptographic strength. Finally, CuMAC provides full authentication with cryptographic strength of 128 bits, which is the same as the compound/aggregate MAC. This way, CuMAC enables a receiver to make a trade-off between (accumulated) cryptographic strength and authentication delay. In some latency-tolerant IoT 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

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	200 %	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
CuMAC/S	7640 bytes	8 %	3.809 ms	128 bits	83.994 ms	128 bits	50.000 ms	128 bits

or rules prescribed by network traffic processing policies.

Most importantly, findings shown in Figure 9a highlight one critical advantageous attribute of CuMAC/S. We observe that CuMAC/S enables the receiver to achieve 128 bits of cryptographic strength for real-time authentication. In other words, for the messages which can be reliably predicted, the receiver achieves the cryptographic strength of the full authentication without any delay (i.e., immediately after the message is received).

Unreliable Communication Channel. 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 evaluate the effect of unreliable communication channels on the MAC schemes in Figure 9b. In the figure, we observe that the packet processing rate in CuMAC and CuMAC/S is equal to that in the truncated MAC. However, the compound/aggregate MAC can enable the processing of a significantly lower number of packets than CuMAC and CuMAC/S. 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 the processing of other packets. For instance, with a typical 10% packet drop rate, the packet processing rate in the compound/aggregate MAC is around 43% which might lead to an unacceptable performance in any typical IoT application.

VII. Implementation Results

Here we discuss the results obtained from a prototype implementation of CuMAC and CuMAC/S on a real car.

A. Details of Prototype Implementation

Figure 10 illustrates the prototype implementation and the setup that was 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. 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 implemented the OBD-II protocol stack. The Arduino UNO board utilizes an Atmel ATmega328P chip,

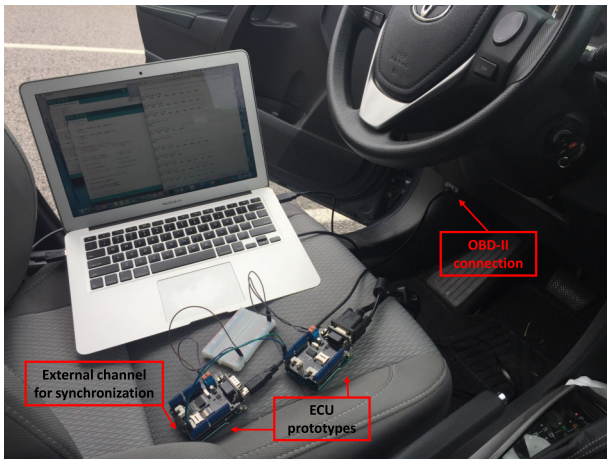


Fig. 10: Prototype connected to a car's CAN bus.

which includes a low-power 8-bit microcontroller 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 [40].

With the above experimental setup, we compared six schemes: the trailing MAC, the truncated MAC, the compound MAC, the aggregate MAC, CuMAC, and CuMAC/S. For all schemes, AES-CMAC with a MAC output of 128 bits was utilized as the underlying MAC algorithm. We utilized an open-source cryptography library [41] to implement AES-CMAC. We found that the average computation time (calculated by averaging the computation time over 1000 executions) of generating a MAC was 0.786 ms. For all MAC schemes except the trailing MAC, 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. Also, we let 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 messages. The table also presents the cryptographic strengths for real-time, full and partially accumulated authentication in each scheme. From Table IV, we observe that: (1) In comparison to other MAC schemes, additional storage for CuMAC and CuMAC/S is at most 2KB, which is acceptable compared to 32KB total flash memory; (2) Unlike the trailing MAC, CuMAC and CuMAC/S do not increase the busload significantly; (3) Unlike the compound MAC and the aggregate MAC, CuMAC and CuMAC/S provide real-time authentication; and (4) In comparison with the truncated MAC, the compound MAC, and the aggregate MAC schemes, CuMAC and CuMAC/S provide significantly higher cryptographic strength for partially accumulated authentication within the processing deadline; (5) CuMAC and CuMAC/S bring at most 0.4 ms extra delay, which is acceptable compared to the case where the service delay on the internet of vehicles can be several seconds [16].

Table IV also shows that compared to CuMAC, CuMAC/S achieves significantly higher cryptographic strength for real-time authentication and partially accumulated authentication at the cost of slightly higher verification delay and extra code space. Hence, in delay-sensitive application scenarios such as CAN, CuMAC/S achieves an advantageous trade-off. Meanwhile, since CuMAC/S shows no advantage over CuMAC for full authentication, CuMAC would be more preferable in delay-insensitive application scenarios such as Sigfox.

VIII. 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 the authentication. We also proposed a variant of CuMAC called *CuMAC with speculation* (CuMAC/S) that is more suitable for latency-sensitive applications. Our promising simulation and experimental results validate that CuMAC and CuMAC/S provide significant advantages over the MAC schemes in the prior art when deployed in emerging IoT applications, including those that run on energy/bandwidth-constrained networks.

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 *IEEE Vehicular Technology Conference (VTC Fall)*, 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] F. Chang and G. A. Gibson, "Automatic I/O hint generation through speculative execution," in *Proceedings of the Third Symposium on Operating Systems Design and Implementation (OSDI)*, 1999, pp. 1–14.

[11] E. B. Nightingale, P. M. Chen, and J. Flinn, "Speculative execution in a distributed file system," in *Proceedings of the Twentieth ACM Symposium on Operating Systems Principles (SOSP)*, 2005, pp. 191–205.

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

[13] 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

[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] A. Rachedi and H. Badis, "Badzak: An hybrid architecture based on virtual backbone and software defined network for internet of vehicles," in *IEEE International Conference on Communications (ICC)*, 2018, pp. 1–7.

[16] T. Mekki, R. Jmal, L. Chaari, I. Jabri, and A. Rachedi, "Vehicular fog resource allocation scheme: A multi-objective optimization based approach," in *IEEE 17th Annual Consumer Communications & Networking Conference (CCNC)*, 2020, pp. 1–6.

[17] T. Mekki, I. Jabri, A. Rachedi, and M. ben Jemaa, "Vehicular cloud networks: Challenges, architectures, and future directions," *Vehicular Communications*, vol. 9, pp. 268–280, 2017.

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

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

[20] 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.

[21] 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.

[22] 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.

[23] 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.

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

[25] G. Bansal, N. Naren, V. Chamola, B. Sikdar, N. Kumar, and M. Guizani, "Lightweight mutual authentication protocol for V2G using physical unclonable function," *IEEE Transactions on Vehicular Technology*, vol. 69, no. 7, pp. 7234–7246, 2020.

[26] V. Chamola, A. Sancheti, S. Chakravarty, N. Kumar, and M. Guizani, "An IoT and edge computing based framework for charge scheduling and EV selection in V2G systems," *IEEE Transactions on Vehicular Technology*, vol. 69, no. 10, pp. 10 569–10 580, 2020.

[27] T. Alladi, S. Chakravarty, V. Chamola, and M. Guizani, "A lightweight authentication and attestation scheme for in-transit vehicles in IoV scenario," *IEEE Transactions on Vehicular Technology*, vol. 69, no. 12, pp. 14 188–14 197, 2020.

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

[29] 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.

[30] 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.

[31] 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.

[32] 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.

[33] 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.

[34] 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.

[35] 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.

[36] 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.

[37] 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.

[38] Ford Motor Company, CrossChasm, and Bug Labs. OpenXC platform. Accessed: July 1, 2019. [Online]. Available: http://openxcplatform.com/resources/traces.html

[39] S. Makridakis and M. Hibon, "Arma models and the box–jenkins methodology," *Journal of Forecasting*, vol. 16, no. 3, pp. 147–163, 1997.

[40] 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.

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

[42] O. Eikemeier, M. Fischlin, J.-F. Götzmann, A. Lehmann, D. Schröder, P. Schröder, and D. Wagner, "History-free aggregate message authentication codes," in *International Conference on Security and Cryptography for Networks*, 2010, pp. 309–328.

APPENDIX A
SECURITY DEFINITION

Here we present the formal security definitions for CuMAC and CuMAC/S. Katz et al. provide the first concrete proof which illustrates that if multiple conventional MACs with cryptographic strength of λ bits are aggregated by XOR operation to form an aggregate MAC, then the aggregate MAC is secure with the cryptographic strength of λ bits [9], [42]. The aggregation procedure employed in CuMAC and CuMAC/S share similar attributes with the scheme proposed by Katz et al. Hence, we present the security definitions which closely follow those presented by Katz et al.

The security evaluation for CuMAC is centered around the notion of unforgeability under chosen message attack with parameter r (uf-cma-r), where r indicates the number of packets accumulated for tag verification. We denote by $\mathbf{Adv}^{\text{uf-cma-}r}_{\mathsf{CuMAC}}(\mathcal{A}, \lambda, q)$, the advantage of the adversary $\mathcal{A}$ in forging a message for a random key $\mathtt{k} \leftarrow \mathsf{KeyGen}(1^\lambda)$, where $\mathcal{A}$ can make q queries to the tag generating oracle of CuMAC $O_{\mathsf{CuMAC}}(\mathtt{k}, \cdot)$, and verification is performed after accumulating r segments of each MAC. CuMAC is considered to be secure if the advantage of the adversary $\mathcal{A}$ is negligibly small. Formally, the advantage can be expressed by the probability (represented by $\Pr[]$) that the following experiment returns 1.

$\mathbf{Exp}^{\text{uf-cma-}r}_{\mathsf{CuMAC}}(\mathcal{A}, \lambda, q)$

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

Invoke $\mathcal{A}^{O_{\mathsf{CuMAC}}(\mathtt{k},\cdot)}$ who can make up to q queries to the tagging oracle of CuMAC $O_{\mathsf{CuMAC}}(\mathtt{k}, \cdot)$. $\mathcal{A}$ can query $O_{\mathsf{CuMAC}}(\mathtt{k}, \cdot)$ with n arbitrarily chosen messages and receive their CuMAC tags in response.

$\mathcal{A}$ outputs a set of n pairs $(\{m_i\}_{i=1}^n, \{\tau_i\}_{i=1}^n)$.

Return 1 if $\mathtt{valid} \leftarrow \mathsf{TagVerify}(\mathtt{k}, i, m_i, \tau_i)$ for all $1 \leq i \leq n$, and $\mathcal{A}$ did not make the query for m_{i^*} to $O_{\mathsf{CuMAC}}(\mathtt{k}, \cdot)$, where $i^* = n - r + 1$.

Return 0 otherwise.

Definition 1. CuMAC is (t, q, ϵ, r)-uf-cma secure if for any probabilistic polynomial time (PPT) adversary $\mathcal{A}$ running in time t, $\Pr\left[\mathbf{Exp}^{\text{uf-cma-}r}_{\mathsf{CuMAC}}(\mathcal{A}, \lambda, q) = 1\right] \leq \epsilon$.

Similar to the experiment $\mathbf{Exp}^{\text{uf-cma-}r}_{\mathsf{CuMAC}}(\mathcal{A}, \lambda, q)$, the uf-cma-$r$ experiment for CuMAC/S can be readily defined as follows.

$\mathbf{Exp}^{\text{uf-cma-}r}_{\mathsf{CuMAC/S}}(\mathcal{A}, \lambda, q)$

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

Invoke $\mathcal{A}^{O_{\mathsf{CuMAC/S}}(\mathtt{k},\cdot)}$ who can make up to q queries to the tagging oracle of CuMAC/S $O_{\mathsf{CuMAC/S}}(\mathtt{k}, \cdot)$. $\mathcal{A}$ can query $O_{\mathsf{CuMAC/S}}(\mathtt{k}, \cdot)$ with $2n - 1$ arbitrarily chosen messages and receive their CuMAC/S tags in response.

$\mathcal{A}$ outputs a set of $2n - 1$ pairs $\left(\{m_i\}_{i=1}^{2n-1}, \{\tau_i\}_{i=1}^{2n-1}\right)$.

Return 1 if $\mathtt{valid} \leftarrow \mathsf{TagVerify}(\mathtt{k}, i, m_i, \tau_i)$ for all $1 \leq i \leq 2n - 1$, and $\mathcal{A}$ did not make the query for m_{i^*} to $O_{\mathsf{CuMAC/S}}(\mathtt{k}, \cdot)$, where $i^* = 2n - r$.

Return 0 otherwise.

Definition 2. CuMAC/S is (t, q, ϵ, r)-uf-cma secure if for any PPT adversary $\mathcal{A}$ running in time t, $\Pr\left[\mathbf{Exp}^{\text{uf-cma-}r}_{\mathsf{CuMAC/S}}(\mathcal{A}, \lambda, q) = 1\right] \leq \epsilon$.

Note that if CuMAC and CuMAC/S are (t, q, ϵ, r)-uf-cma secure for all r, then they are also (t, q, ϵ)-uf-cma secure which is the standard notion of security for a MAC scheme. We utilize the aforementioned uf-cma-r security model to define the cryptographic strength for full authentication, partially accumulated authentication, and real-time authentication. Note that in the uf-cma-r experiment, when the experiment returns a value of 1, it implies that $\mathcal{A}$ can forge a valid tag for a packet at which point the receiver has already accumulated r packets. Therefore, if CuMAC or CuMAC/S is (t, q, ϵ, n) secure, i.e., $r = n$, then CuMAC or CuMAC/S is secure in terms of full authentication. Similarly, if CuMAC or CuMAC/S is (t, q, ϵ, r) secure for all $2 \leq r \leq n - 1$, then the scheme is secure for partially accumulated authentication; and if CuMAC or CuMAC/S is $(t, q, \epsilon, 1)$ secure, i.e., $r = 1$, then the scheme is secure in terms of real-time authentication.

The security of CuMAC and CuMAC/S is based on the following assumption that defines the security of the underlying MAC algorithm [37].

Assumption 1. *The underlying deterministic MAC algorithm,* MacGen*, is (t, q, ϵ)-uf-cma secure—i.e., the probability that an adversary will be successful in producing a forged tag after running for a polynomial-time t and making q queries is negligible.*

APPENDIX B
SECURITY PROOF

Here we present theorems and corresponding proofs for the security for CuMAC and CuMAC/S. Let CuMAC be instantiated with parameters (l, n), i.e., each MAC is divided into n segments, each of length l bits. Let CuMAC/S be instantiated with parameters (β, l, n), i.e., the speculation error rate for the messages is β, and each MAC is divided into n segments, each of length l bits. Note that in CuMAC and CuMAC/S, the receiver performs real-time authentication by setting $r = 1$, partially accumulated authentication by setting $1 < r < n$, and full authentication by setting $r = n$.

Theorem 1. *For any $t, q \in \mathbb{N}$ and $\epsilon > 0$, if the underlying deterministic MAC algorithm,* MacGen*, is (t, q, ϵ)-uf-cma secure, then CuMAC with parameters (l, n) is (t', q', ϵ', r)-uf-cma secure, where*

$$t' \approx t, \quad q' = \frac{q - n + 1}{n}, \quad \epsilon' = 2^{l(n-r)} \cdot \epsilon.$$

Proof: Let there be an adversary $\mathcal{A}$ that succeeds to create a forgery of an authentication tag for CuMAC with a non-negligible probability. We construct a simulator $\mathcal{S}$ that interacts with the adversary $\mathcal{A}$ and creates a forgery of a MAC for the MacGen algorithm with a non-negligible probability.

Let CuMAC and the MacGen algorithm utilize the same secret key $\mathtt{k}$ which is not known to the adversary $\mathcal{A}$. Also, let the MAC of a message in CuMAC be computed by a query to the tag generating oracle of underlying MAC, which is denoted as $O_{\mathsf{MacGen}}(\mathtt{k}, \cdot)$. In this way, $\mathcal{S}$ perfectly simulates $O_{\mathsf{CuMAC}}(\mathtt{k}, \cdot)$, and hence, the uf-cma-$r$ experiment. Suppose the uf-cma-r experiment for CuMAC returns 1 with the probability ϵ' in time t', where an adversary $\mathcal{A}$ outputs a valid forgery $(\{m_i\}_{i=1}^n, \{\tau_i\}_{i=1}^n)$ after q' queries to $O_{\mathsf{CuMAC}}(\mathtt{k}, \cdot)$ simulated by $\mathcal{S}$. To create a forgery of a MAC for the MacGen algorithm, the simulator $\mathcal{S}$ proceeds as follows.

For all $i \in [1, n]$ and $i \neq i^*$, the simulator $\mathcal{S}$ queries the $O_{\mathsf{MacGen}}(\mathtt{k}, \cdot)$ for the MAC of m_i, and obtains the corresponding σ_i. It divides each MAC into n segments as shown in equation (1). It recovers the MAC segments of the message m_{i^*} by removing the mask by the MAC segments of other messages as follows:

$$s_{i^*}^k \leftarrow \tau_{i^*+k-1} \oplus \bigoplus_{j=1, j\neq k}^{n} s_{i^*+k-j}^j. \tag{4}$$

Since $i^* = n - r + 1$, the simulator $\mathcal{S}$ cannot recover the segments $s_{i^*}^k$ with $k \geq r + 1$. Hence, it makes a random guess for the rest of the $n - r$ segments, such that $\widetilde{s}_{i^*}^k \leftarrow\$ \{0,1\}^l$ for all $k \in [r + 1, n]$. Finally, to create the forgery for the underlying MAC algorithm, MacGen, it concatenates all the recovered segments and the guessed segments: $\sigma_{i^*} \leftarrow s_{i^*}^1 || s_{i^*}^2 \cdots || s_{i^*}^r || \widetilde{s}_{i^*}^{r+1} || \cdots \widetilde{s}_{i^*}^n$. This means that given a successful forgery of the authentication tag in CuMAC, the probability of creating the forgery of MacGen is $2^{-l(n-r)}$.

To achieve the forgery of MacGen as shown above, the simulator $\mathcal{S}$ conducts at most $n \cdot q'$ queries to the $O_{\mathsf{MacGen}}(\mathtt{k}, \cdot)$ to reply the q' queries by $\mathcal{A}$ to $O_{\mathsf{CuMAC}}(\mathtt{k}, \cdot)$. Also, the simulator $\mathcal{S}$ conducts $n - 1$ queries to $O_{\mathsf{MacGen}}(\mathtt{k}, \cdot)$ to obtain $\{\tau_i\}_{i=1, i\neq i^*}^n$. Therefore, if these exists an adversary $\mathcal{A}$ running in time t' and achieving $\Pr\left[\mathbf{Exp}_{\mathsf{CuMAC}}^{\text{uf-cma-}r}(\mathcal{A}, \lambda, q') = 1\right] \leq \epsilon'$, then it can be leveraged to create a forgery for the underlying MAC algorithm, MacGen, in time t' plus the time required to evaluate the equation (4), by making $nq' + n - 1$ queries, and with probability $2^{-l(n-r)}\epsilon'$. Hence, if the underlying MAC algorithm, MacGen, is (t, q, ϵ)-uf-cma secure, then CuMAC is (t', q', ϵ', r)-uf-cma secure, where $t' \approx t$, $q' = \frac{q-n+1}{n}$, and $\epsilon' = 2^{l(n-r)}\epsilon$. ∎

Theorem 2. *For any $t, q \in \mathbb{N}$ and $\epsilon > 0$, if the underlying deterministic MAC algorithm,* MacGen*, is (t, q, ϵ)-uf-cma secure, then CuMAC/S with parameters (β, l, n) is (t', q', ϵ', r)-uf-cma secure, where*

$$t' \approx t, \quad q' = \frac{q - 3n + 3}{2n - 1}, \quad \epsilon' = \frac{\epsilon}{(1 - \beta) + \beta 2^{-l(n-r)}}.$$

Proof: Let there be an adversary $\mathcal{A}$ that succeeds to create a forgery of an authentication tag for CuMAC/S with a non-negligible probability. We construct a simulator $\mathcal{S}$ that interacts with the adversary $\mathcal{A}$ and creates a forgery of a MAC for the MacGen algorithm with a non-negligible probability.

Let CuMAC/S and the MacGen algorithm utilize the same secret key $\mathtt{k}$ which is unknown to the adversary $\mathcal{A}$. Also, let the MAC of a message in CuMAC/S be computed by a query to $O_{\mathsf{MacGen}}(\mathtt{k}, \cdot)$. In this way, $\mathcal{S}$ perfectly simulates $O_{\mathsf{CuMAC/S}}(\mathtt{k}, \cdot)$, and hence the uf-cma-$r$ experiment for CuMAC/S. Suppose the uf-cma-r experiment for CuMAC/S returns 1 with the probability ϵ' in time t', where an adversary $\mathcal{A}$ outputs a successful forgery $\left(\{m_i\}_{i=1}^{2n-1}, \{\tau_i\}_{i=1}^{2n-1}\right)$ after q' queries to $O_{\mathsf{CuMAC/S}}(\mathtt{k}, \cdot)$ simulated by $\mathcal{S}$. To create a forgery of a MAC for the MacGen algorithm, the simulator $\mathcal{S}$ proceeds as follows.

For all $i \in [1, 2n - 1]$ and $i \neq i^*$, the simulator $\mathcal{S}$ queries $O_{\mathsf{MacGen}}(\mathtt{k}, \cdot)$ for the MAC of m_i, and obtains the corresponding σ_i. Additionally, it queries $O_{\mathsf{MacGen}}(\mathtt{k}, \cdot)$ for the MAC of the speculated messages $\widehat{m}_i$ and obtains $\widehat{\sigma}_i$ for all $i \in [i^* + 1, i^* + n - 1]$. It divides each MAC into n segments as shown in equation (1). It recovers the MAC segments of the message m_{i^*} by removing the mask by the MAC segments of other messages as follows:

$$s_{i^*}^k \leftarrow \tau_{i^*+k-1} \oplus \bigoplus_{j=1, j\neq k}^{n} s_{i^*-j+k}^j \oplus \bigoplus_{j=2}^{n} \widehat{s}_{i^*+j+k-2}^j. \tag{5}$$

By following the above procedure, the simulator $\mathcal{S}$ recovers r MAC segments. For all $k \geq r+1$, the simulator $\mathcal{S}$ attempts to recover $s_{i^*}^k$ from the tags received before tag τ_{i^*} as follows:

$$s_{i^*}^k \leftarrow \tau_{i^*-k+1} \oplus \bigoplus_{j=1}^{n} s_{i^*-k-j+2}^j \oplus \bigoplus_{j=2, j\neq k}^{n} \widehat{s}_{i^*-k+j}^j. \tag{6}$$

These segments can be recovered with a probability $1-\beta$. If a speculation error occurs, then the corresponding MAC segment is not recovered. In this case, $\mathcal{S}$ sets the value of the MAC segment by randomly guessing the bits. Finally, the simulator $\mathcal{S}$ creates a fresh forgery for the underlying deterministic MAC algorithm, MacGen, by concatenating all recovered and guessed segments. The probability that such forgery is correct is $(1-\beta)+\beta\cdot 2^{-l(n-r)}$.

To achieve the forgery of MacGen as shown above, the simulator $\mathcal{S}$ conducts at most $(2n-1)q'$ queries to $O_{\mathsf{MacGen}}(\mathtt{k},\cdot)$ to answer q' queries by $\mathcal{A}$ to $O_{\mathsf{CuMAC/S}}(\mathtt{k},\cdot)$. In order to compute operations in equations (5) and (6), the simulator $\mathcal{S}$ conducts at most $2n-2$ queries to $O_{\mathsf{MacGen}}(\mathtt{k},\cdot)$ to obtain $\{\tau_i\}_{i=1,i\neq i^*}^{2n-1}$, and at most $n-1$ queries to obtain $\{\widehat{\sigma}_i\}_{i=i^*+1}^{i^*+n-1}$. Therefore, if for an adversary $\mathcal{A}$ running in time t', we have $\Pr\left[\mathbf{Exp}^{\text{uf-cma-}r}_{\mathsf{CuMAC/S}}(\mathcal{A},\lambda,q')=1\right]\leq\epsilon'$, then we can leverage it to break the underlying MAC algorithm, MacGen, in time t' plus the time required to evaluate equations (5) and (6), by making $(2n-1)q'+3n-3$ queries, and with probability $\epsilon'(1-\beta+\beta 2^{-l(n-r)})$. Hence, if the underlying MAC algorithm, MacGen, is (t,q,ϵ)-uf-cma secure, then CuMAC/S is (t',q',ϵ',r)-uf-cma secure, where $t'\approx t$, $q'=\frac{q-3n+3}{2n-1}$, and $\epsilon'=\frac{\epsilon}{1-\beta+\beta 2^{-l(n-r)}}$. ■