A Resource-Preserving Self-Regulating Uncoupled MAC Algorithm to be Applied in Incident Detection

Michael Heigl[a,b], Laurin Doerr[a,b], Nicolas Tiefnig[b], Dalibor Fiala[a], Martin Schramm[b]

[a] *University of West Bohemia, Pilsen, Czech Republic*
[b] *Deggendorf Institute of Technology, Deggendorf, Germany*

Abstract

The connectivity of embedded systems is increasing accompanied with thriving technology such as Internet of Things/Everything (IoT/E), Connected Cars, Smart Cities, Industry 4.0, 5G or Software-Defined Everything. Apart from the benefits of these trends, the continuous networking offers hackers a broad spectrum of attack vectors. The identification of attacks or unknown behavior through Intrusion Detection Systems (IDS) has established itself as a conducive and mandatory mechanism apart from the protection by cryptographic schemes in a holistic security eco-system. In systems where resources are valuable goods and stand in contrast to the ever increasing amount of network traffic, sampling has become a useful utility in order to detect malicious activities on a manageable amount of data. In this work an algorithm - Uncoupled MAC - is presented which secures network communication through a cryptographic scheme by uncoupled Message Authentication Codes (MAC) but as a side effect also provides IDS functionality producing alarms based on the violation of Uncoupled MAC values. Through a novel self-regulation extension, the algorithm adapts it's sampling parameters based on the detection of malicious actions. The evaluation in a virtualized environment clearly shows that the detection rate increases over runtime for different attack scenarios. Those even cover scenarios in which

Email addresses: `heigl@kiv.zcu.cz, michael.heigl@th-deg.de` (Michael Heigl), `laurind@kiv.zcu.cz, laurin.doerr@th-deg.de` (Laurin Doerr), `nicolas.tiefnig@th-deg.de` (Nicolas Tiefnig), `dalfia@kiv.zcu.cz` (Dalibor Fiala), `martin.schramm@th-deg.de` (Martin Schramm)

intelligent attackers try to exploit the downsides of sampling.

Keywords: network security, adaptive intrusion detection, message authentication, self-regulation, resource conservation

1. Introduction

Many areas such as automotive or industrial are pervaded by various trends. Whereas these fields in the past have been isolated and permeated by proprietary technology, recent developments tend towards unified concepts and mechanisms boosted by new technologies. Internet of Things (IoT) will further accelerate this development and enables networking and control of components across the existing network infrastructure. Thereby also computer-aided systems are better integrated in the physical environment. Ubiquitous connectivity creates a complex network of things that includes vehicles, but also smart infrastructure, buildings and smart homes. With the rise and success of cryptocurrencies even a possibility for micro- and nanopayments between machines was created. The advent of such Distributed Ledger Technologies, for instance IOTA, will further not only benefit the rapidly developing IoT industry [1] but also connected mobility in which vehicles turn into digital platforms. For example, with intelligent transportation systems taking advantage of features and capabilities from Wireless Sensor Networks (WSN) a bridge to IoT-enabled Smart Cars is built [2].

However, accompanied with these trends, the too fast development, rapidly advancing technologies and pervasion of multiple areas lead also to a broad range of security problems considering the incrementation of hacker abilities such as the usage of cloud or distributed computing, quantum computation, etc. [3]. The application of various protocols and devices with different resource requirements (bandwidth, computation) often accompanied in deterministic timing environments by increasing traffic amounts makes a holistic security solution difficult especially when outdated legacy components are still participating in the network. What is more, the preservation of computing resources, e.g. for IoT-enabled devices, stands in conflict with the resource requirements of detection mechanisms when applied in high-volume networks.

A number of techniques have been designed for such environments protecting against and detecting attacks through cryptographic mechanisms or Intrusion Detection Systems (IDS) [4]. The latter is an efficient possibility to detect malicious behavior even when cryptography is broken [5]. Nevertheless, for a comprehensive security solution a defense-in-depth concept must include cryptographic procedures as well as IDS components [3]. In addition, applied sampling methods in security mechanisms need more attention. The advent of pervasive computing will not only increase the amount of data in cloud but also in embedded environments such as for smart sensors in automobiles struggling with increasing data traffic. Thus, sampling could be an efficient countermeasure for devices having only limited processing capabilities.

In this work an uncoupled Message Authentication Code (MAC) algorithm called Uncoupled MAC is extended to work as an IDS generating alarms if the authenticity and integrity of network communication get violated. This benefits especially resource-constrained environments where protection goals must be guaranteed and cryptographic schemes are necessary. With respect to the lightweight and resource-aware security concept of [6], Uncoupled MAC even provides simple IDS capability on a resource-aware cryptographic layer. No additional complex IDS, providing e.g. behavioral-based detection by machine learning algorithms, must be applied if they are not feasible due to energy limitations and static communication [7]. However, Uncoupled MAC is not a replacement for a sophisticated and powerful IDS solution. While only monitoring a predefined ratio of sampled packets, a decent overhead on computational resources as well as network traffic can be preserved. By introducing a self-regulation extension, the Uncoupled MAC parameters defining its' sampling mechanism can be automatically adjusted in a dynamic manner according to the detection of Uncoupled MAC violations.

The rest of this paper is structured as follows: Section 2 provides an overview on incident detection mechanisms ranging from classical IDSs to the exploitation of the nature of cryptographic mechanisms in order to detect security-relevant incidents. Furthermore sampling systems are presented including Uncoupled MAC. Improvements and extensions of the original Uncoupled MAC algorithm are presented in Section 3. The novel self-regulating sampling approach based on the detection of Uncoupled MAC violations is described in Section 4. In Section 5 details on the implementation and evaluation environment are presented, attack scenarios described and evaluation metrics provided. Results of several measurements mainly showing the advantage of the self-regulation compared to the non-self-regulated version of Uncoupled MAC by also discussing the trade-off between detection capability and resource utilization are given in Section 6. A short conclusion and a glance at the future work of the ongoing research work is finalizing the article in Section 7.

2. Incident Detection Mechanisms

Incident detection is part of an incident handling process in which malicious behavior for instance in networks can be detected, analyzed and appropriate reaction mechanisms can be planned and executed. In Fig. 1 the process with involved components is shown. After detecting anomalous or intrusive behavior through the *Incident Detection* module, the resulting alarms will be logged or visualized within the *Incident Post-Processing* module and analyzed for a later reaction by the *Incident Response* module in the *Incident Analysis/Control* module. Incident analysis techniques for instance could comprise performing correlation or similarity functions on raised alerts. Based on the intelligence of this module, a proper reaction to the detected intrusion or anomaly can be planned and executed. Such a reaction could include reconfiguring the network through Software-Defined Networking (SDN) techniques, generating new configurations for firewalls, creating new misuse-based IDS rules or adapting the parameters for other incident detection mechanisms. Apart from the mentioned functions of the *Incident Post-Processing* module, it can further be used for monitoring other incident handling components.

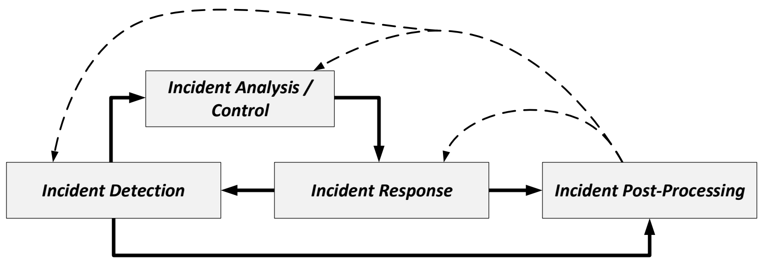

Fig. 1. Incident handling process

To defend against various types of cyber attacks, there are two main streams of security solutions: cryptographic schemes and intrusion detection. Although state-of-the-art IDSs are capable of detecting some basic attacks such as fabrication attack and suspension attack, they fail to detect more sophisticated ones such as the masquerade attack [8]. The following described IDSs together with cryptographic mechanisms for network protection are integral parts in modern IT security eco-systems and serve as a fundamental basis for the detection of security incidents.

2.1. Classical IDS for Incident Detection

IDSs are used to monitor, detect and analyze events that are considered as violation to the security policies of a networked environment [9]. They can detect malicious activities or policy violations by monitoring the network traffic or system activity [10]. An IDS is normally a stand-by device or third-party software which will not inquire many changes to the current system. It is suitable for resource-constrained or inherited systems to protect their network security [11].

There are three main architectures that IDSs can be divided into: Host-based IDSs (HIDS), Application-based IDSs (AIDS) and Network-based IDSs (NIDS). The former exists of an agent which recognizes intrusions on a host/machine by analyzing internal operations for instance system calls or illegitimate resource utilization. An example of a distributed denial-of-service attack detection in cloud computing by applying a HIDS is presented in [12]. An AIDS is presented in [13] which collects, analyzes and detects intrusion from application-specific data from a single application, e.g. a webserver log-file. These architectures have not received much attention in recent years since they have significant drawbacks compared to NIDS [14].

Network-based systems can be placed at dedicated points within a network (firewall, router, network host) and detect malicious activities based on network data. The following detection methods can be applied regardless of the architecture HIDS, AIDS or NIDS. The techniques of these methods can be based for instance on statistics, specifications or machine learning approaches.

Misuse-based IDS

This method, also called Signature- or Knowledge-based, is based on a set of rules or patterns which are either pre-configured by the system or manually by an administrator. These signatures describe network attacks and trigger alerts if they detect those. However, this approach does not contribute much in terms of zero-day attack detection. The main drawback is how to build permanent signatures that have all the possible variations and non-intrusive activities in network environments [15].

Anomaly-based IDS

This method collects data containing examples of normal behavior and builds a model of familiarity, therefore, any action that deviates from the model is considered suspicious and is classified as an intrusion [16]. Anomaly-based systems are able to detect zero-day intrusions but suffer from a higher possibility for false negative and false positive alarms.

To overcome the drawbacks and exploit the advantages of both methods (misuse- and anomaly-based), hybrid systems are proposed, e.g. in [17].

Adaptive IDS

The term adaptive for IDS in the context of this work describes techniques of sampling and self-regulation. To cope with the increasing amount of traffic within networks while reducing large memory and CPU processing requirements, sampling turned out to be a promising scalable data aggregation technology for IDSs since the processing capacity of such systems are typically much smaller than the amount of data to be inspected. Because sampled traffic is an incomplete approximation of the actual one, multiple mechanisms for sampling data exist.

Sampling. In [18] a difference between packet- and flow-based sampling, crucial for the working of NIDS, together with deterministic and non-deterministic methods is made. Packet-based sampling is simple to implement with low CPU power and memory requirements but is inaccurate for inference of flow statistics like size distribution of the original flows. In contrast, flow-based sampling overcomes the limitations of packet-based sampling but suffers from prohibitive memory and CPU power requirements and is still too complex to implement [19]. The flow-size based sampling technique in [18] assumes that network attacks usually use small flows as traffic source. With the proposed selective sampling strategy such flows are sampled with a constant probability. Other related work evaluated that packet sampling has a negative impact on the efficiency of anomaly-based IDSs increasing the false positives but performs best when using a random flow sampling strategy. However, it is possible to maintain a high level of security while selectively inspecting packets with a minimal amount of processing overhead. An analytic and statistical model for the process of network intrusions has been introduced in [20] supporting the experimental results of [18] demonstrating that it is sufficient to inspect only a small number of sampled packets. In [21] a packet- and time-driven traffic sampling strategy for an IDS in a SDN is proposed that fully utilizes the inspection capability of malicious traffic, while maintaining the total aggregate volume of the sampled traffic below the inspection processing capacity of the IDS. The packet-driven approach inspects a packet every $1/x$ packets for a sampling rate x and the time-driven inspects all the packets within a time window of sampling rate x each sampling interval. The time-driven mechanism has the advantage of detecting stateful attacks because it captures all the packets for a certain time duration. However, if packets are mainly sent event-triggered, the time-driven approach is not feasible since there could be phases of sampling in which no packets are inspected. This could easily happen in networks with high fluctuations of the bandwidth. If an intruder is able to compromise the IDS or might know the sampling rate, he could exploit this knowledge by performing malicious activities outside the sampling interval. By increasing a sampled injection of malicious packets, he could also extract the sampling rate information by observing the reaction of the IDS in a trial and error fashion. Thus, a combination of a packet- and time-driven mechanism could mitigate such problems by applying a random chosen sampling interval within fixed boundaries.

Self-Regulation. Self-regulating sampling mechanisms have been presented in [22], for instance a method managing the processor usage in a network device through adaptive sampling in network security applications. The authors state that a wide range of common network anomalies only require a single sample in order to provide 100% accuracy of detection but there are also other network anomalies which cannot be detected with a single sample. An example is an anomaly which misuses a protocol for purposes which were not meant for it. This requires more advanced techniques than a simple signature check. Cryptographic mechanisms could be used to overcome such limitations. In [23] an adaptive packet-level sampling method on different traffic fluctuations and burst scales has been introduced. The method can dynamically adjust each packet sampling probability depending on the magnitude of traffic fluctuation. This approach achieves higher accuracy in contrast to random sampling methods. Another adaptive sampling method for anomaly detection algorithms has been presented in [24]. The adaptive sampling described is a promising general sampling technique that preserves well the traffic feature distributions and at the same time is able to improve the detection capabilities of the system. A hybrid sampling algorithm combining both flow statistics and feedback to intelligently choose the packets to sample is presented in [25] in order to achieve self-regulation. The sampling rate is determined by the current workload in the cloud, and thus minimizing the effects to normal workload. By the CIDS framework defined, an off-the-shelf IDS can be utilized in a cloud environment by reducing and balancing the data collection (packet capturing, filtering, sampling rate) and computation workload dynamically according to the resource utilizations in the cloud. Another example of adaptive sampling systems is the work presented in [26] that aims to effectively reduce the volume of traffic that Peer-to-Peer (P2P) botnet detectors need to process while not degrading their detection accuracy. The system first identifies a small number of potential P2P in high-speed networks for botnet detection. In a 2-step approach first a suspicious host identification is performed by roughly sampling the traffic in order to detect potential P2P bots quickly. Second an in-depth analysis with more fine-grained detectors achieve an accurate detection on the identified hosts.

Applying sampling techniques in conjunction with a self-regulating IDS helps to reduce the measurement overhead for an IDS in terms of CPU, memory or bandwidth enabling the application of a partial IDS in future connected embedded systems. Similar to the concept of partial networking, the IDS components regulate their activeness such that in times of higher detection of malicious actions within the network more packets will be sampled leading to a higher resource consumption. On the other hand, in times of less or no detection, the IDS components lower their sampling or might even partially turn off completely. Furthermore, by using adaptive techniques that regulate, for instance, IDS relevant parameters or the sampling rate, the security level of a system can be adjusted by preserving a controllable overhead on resources.

2.2. Cryptographic Mechanisms for Incident Detection

In contrast to the classical IDS based on e.g. statistical techniques, the following mechanisms are based on cryptographic schemes that can be exploited for alert generation and thus provide incident detection capability. Apart from the functionality of cryptographic protection, alerts can be raised if characteristics of the mechanisms fail, e.g. drifting sequence counters, incorrect decryption of messages or mismatching hash digests.

Encryption Schemes

To the best of the authors knowledge no work could be found during the writing of this article combining encryption schemes with IDS functionality and sampling. However, one can distinguish between two types of secure data: full encryption and selective encryption. Full encryption means that all transmitted data is secured by cryptographic methods. These include the classic encryption protocols on Ethernet such as MACsec, IPsec, TLS or SSH. Similar to the sampling for IDSs, selective encryption mechanisms could be found mainly targeted to secure the data transfer over Ethernet on several communication layers.

MACsec, for instance, provides hop-by-hop security for layer-2 of Ethernet and is compatible with most of the Ethernet-based protocols. The data transmission is secured with GCM-AES. Within the MACsec frame format a Security Tag as an extension of the EtherType is defined that contains information about the association number within the channel, a packet number for replay protection and to provide a unique initialization vector for encryption and authentication algorithms which helps the receiver to identify the decryption key. The Integrity Check Value (ICV) generated by the GCM-AES following the payload guarantees that the packet was created by a node possessing the correct key. In the case of mismatching ICVs the frame gets dropped. This circumstance, failures in decryption or mismatching packet numbers could be exploited to trigger alerts. The same applies to other encryption protocols, for instance when HMACs in IPsec do not match. However, since each device on a transfer route needs an implementation of the MACsec standard including special PHYs on all physical interfaces, MACsec is not very common and there are not many devices on the market. All components on a data path which have to be secured need certificates or mechanisms for pre-shared keys. Other protocols, according to [27], such as IPsec or TLS are very complex, suffer from context specific attacks, do not provide layer-2 security, or suffer from the lack of interoperability between their different versions.

In the context of selective encryption on the one side, the encryption of certain portions of a message is understood reducing the overhead spent on the encryption/decryption process. On the other side, selective encryption indicates that a sufficient number of messages is encrypted providing the necessary confidentiality for message transmission. This technique is mainly applied in resource-aware environments such as WSN or Mobile Ad Hoc Networks. Two methods are proposed in [28], alongside with a comprehensive literature survey, for the selection of message encryption. The first approach encrypts all messages and the second is based on the toss-a-coin approach in which approximately a half of the data get encrypted. The selective approach, however, leaves nearly a 50% chance for an adversary open to inject or manipulate packets which get not encrypted. The main application for selective encryption is in the field of image and video streaming and was not defined for the utilization on event-triggered messages. Further, the application in legacy-systems with legacy protocols is difficult since the encryption and decryption process retrofitted in these systems adds a significant delay to the original data transmission which makes it hardly applicable for real-time critical protocols. Beyond that, failing encryption or decryption could lead to a severe financial damage for instance in industrial control applications since packets are not guaranteed to be delivered in their deterministic time window. Since confidentiality in embedded environments is often of a minor priority, encryption/decryption schemes could be neglected [29]. Hence, authentication processes might be sufficient for the most of the cases.

Message Authentication Codes

Selective authentication is similar to selective encryption. However, to the best of the authors knowledge no comparable work could be found proposing selective authentication for network packet security. Unlike the full authentication, selective authentication only helps to identify whether or not data is still useful after being modified. A selective authentication approach presented in

[30] is based on semi-fragile watermarks for vector geographical data.

Apart from the consideration of detection and protection methods separately, the authors of [31] and [32] combine the two classes for network security. In [31] a framework is proposed in which a multimodal biometrics or HMAC scheme is used for continuous authentication. Intrusion detection is modeled as sensors to detect the system's security state in order to protect high security mobile ad hoc networks. Intrusion detection in [31] is modeled as noisy sensors that can detect the system's security state (safe or compromised). Apart from Ethernet-based networks the approach of combining intrusion detection with a protection method is even proposed for the automotive CAN-bus. In [32] a simple intrusion detection method for detecting malicious frames is presented. The basis for this approach is that each electronic control unit is legitimately equipped with a hardware security module. By applying a certain type of MAC (HMAC, AES-CMAC, ...) and a counter within the proposed Domain_Activation and Domain_Violation frames, Denial of Service, replay and impersonation attacks can be detected and the driver notified about the risks.

Uncoupled MAC

Uncoupled MAC was originally described in [33] for securing network communication between legacy devices communicating with protocols which do not provide any security mechanisms. According to [34] and [35] many protocols in the TCP/IP stack do not provide mechanisms to authenticate the source or destination of a network traffic packet enabling to spoof the source address which is used in many attacks. By introducing dedicated embedded plug-in devices which are located interconnected to the legacy devices, authenticity and integrity of data communication can be provided by a MAC uncoupled from the original message.

Thus, for instance with respect to Fig. 2, if legacy device A is sending messages to B ($m_{\mathrm{A}\to\mathrm{B}}$), the embedded plug-in devices D_A and D_B are transparently forwarding the original message but decapsulated from the original message D_A and D_B are generating a MAC ($MAC(m_{\mathrm{A}\to\mathrm{B}})$). D_B is storing its generated

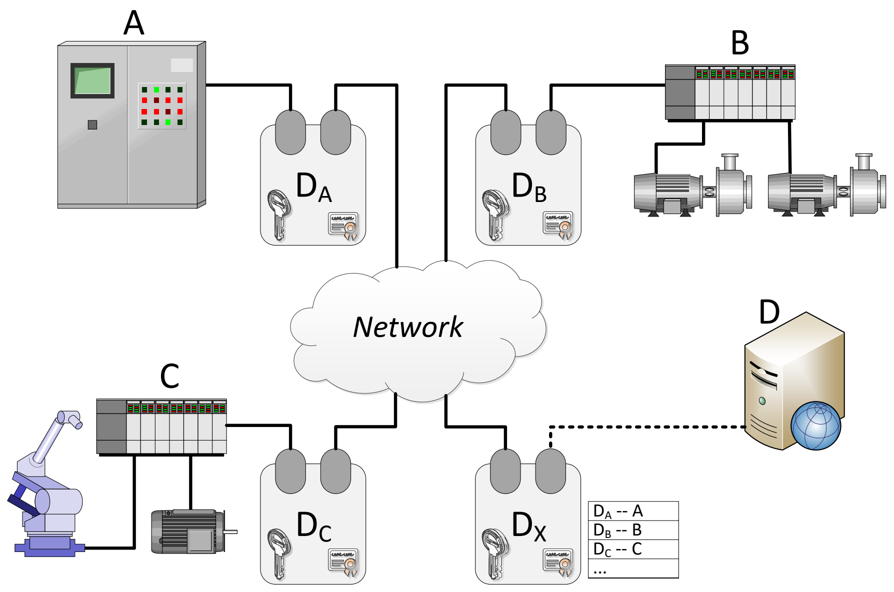

Fig. 2. Network scenario of the proposed concept [36]

MAC and waiting to verify it with the MAC generated by D_A sent over a secure communication channel. If the MACs are identical, data is accepted; otherwise, data is tampered. The overlay communication channel does not influence or delay the original traffic and, thus, prevents failures for instance through the MAC generation. Furthermore, in [33], the proposed security concepts (authenticated boot, direct attestation, key establishment) and the internal architecture for MAC generation and verification have been described for the embedded plug-in devices.

A more comprehensive approach of Uncoupled MAC was described in [36]. By introducing a further embedded surveillance device D_X (Fig. 2) it is possible to securely bring new embedded plug-in devices into the network within the so called *Setup Phase* such that they can participate in the MAC generation and verification process. Within the *Runtime Phase*, D_X is used for aggregating info, warning and error messages produced by the embedded plug-in devices.

A further significant characteristic of Uncoupled MAC is the separation of the Runtime Phase into a *MAC Phase* and an *Idle Phase* with a random duration such that an adversary is not able to exploit the algorithm. By only generating and verifying MACs within the MAC Phase, embedded plug-in devices only use a certain amount of bandwidth in order to have a low and deterministic impact on the original traffic (probe effect). By the predefined boundaries for the random chosen number of packets n, defining the MAC Phase and the duration of the Idle Phase α, a reasonable degree of security and additional overhead can be adjusted. A combination of a packet- and time-driven mechanism mitigates the problems of other systems described in the above sections in which n packets are sampled in each time-driven interval α by applying randomly chosen values within fixed boundaries. This combination allows the protection of event- and time-triggered communication.

Further, in [36] the extension of the MAC packets with a sequence counter and a timestamp was introduced guaranteeing the freshness and order of packets preventing replay attacks. Uncoupled MAC violations against the mentioned extensions are stored as anomaly logs within each individual embedded plug-in device and sent to the surveillance device. Besides a theoretical safety and security assessment, the concept of a complementary applied IDS within the *Control & Management* module is described. Thus, the combination of a protection- and detection-based technique was originally considered.

In [29] an assessment simulation model for a simple scenario including Uncoupled MAC is proposed. With the model, one is able to simulate bandwidth utilization and Uncoupled MAC overhead as well as detection rates by examining different values for n and α for a scenario in which an adversary is randomly injecting packets.

Compared to the approaches from the sections above, Uncoupled MAC has the following advantages and drawbacks.

Advantages

- Plug-and-play capability for legacy devices in form of embedded plug-in devices

- Integration in networked devices ensured by the concept
- Retrofitted cryptographic protection for protocols without security mechanisms
- No influence on underlying communication (e.g. real-time aspect will be preserved)
- Applicability to event- and time-triggered communication by the combination of a packet- and time-driven mechanism
- Functioning in communication intense systems by sampling (e.g. in switches or cloud platforms)
- Adjustable security level and additional overhead by adapting Uncoupled MAC parameters (e.g. in bandwidth-critical systems)
- Combination of a MAC algorithm with an IDS functionality mitigates the lack of data integrity and data origin authenticity as with an IDS alone

Disadvantages

- Overhead on resources such as CPU, memory and bandwidth (but adjustable)
- 100% detection rate cannot be achieved due to sampling

For a holistic security eco-system, a hybrid system existing of the methods shown in Fig. 3 is recommended. However, the main focus of this work is towards the resource conservation and protection methods. The detection by utilizing an IDS has already been proposed in [36]. More of interest is the interaction of the protection by Uncoupled MAC together with a self-regulating sampling method and the generation of alerts. On top of that a typical (more powerful) misuse- or anomaly-based IDS could be employed that is taking the sampled traffic as an input for in-depth intrusion detection. An alert analysis technique such as proposed in [37], on top of the resulting alerts from the protection and detection method, could be used to find a consensus and to initiate further reaction mechanisms, e.g. the reconfiguration or isolation of a network segment through SDN. The proper consensus finding might be beneficial in safety critical environments since it can be used to reduce for instance false positives. Uncoupled MAC allows to filter and sample for a certain protocol on a specific communication flow in parallel for multiple instances. Through the verification of n packets within a MAC Phase, not only packet- but also flow-based features for a downstream applied IDS can be derived.

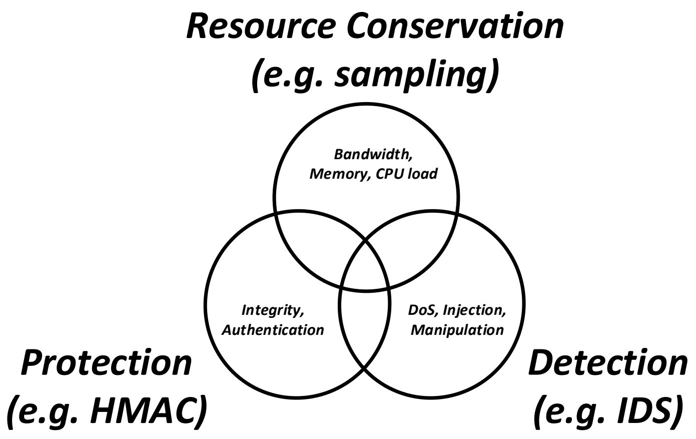
Resource Conservation (e.g. sampling)
Bandwidth, Memory, CPU load
Integrity, Authentication
DoS, Injection, Manipulation
Protection (e.g. HMAC)
Detection (e.g. IDS)

Fig. 3. Combination of methods

3. Uncoupled MAC Algorithm Improvements

In this section improvements compared to the Uncoupled MAC algorithm proposed in [29, 33, 36] are provided. Some of them are needed to enable an IDS functionality of the Uncoupled MAC algorithm and therefore facilitate the interaction with a typical IDS. Other improvements benefit the algorithm for the conservation of resources, with a specifiable level of security, or makes it applicable in a broader sphere and more effective.

3.1. Master/Slave Negotiation

One change of the algorithm compared to the original design, proposed in [36], is the negotiation between two communicating Uncoupled MAC partners devices to generate or verify MACs. Instead of a more complex negotiation between the securely communicating parties, the Uncoupled MAC partner that originally initiated their communication becomes the master and the other one becomes the slave. The direction of the data arriving first at one partner dictates which partner is set to master.

Rather than negotiating between master and slave which device starts with the generation/verification also defining the duration of the Idle Phase and how many packets will be examined during the MAC Phase, only the master is specifying this information and is communicating this to the slave. This significantly increases the performance of the algorithm since the overhead of the negotiation is prevented but the security aspect of randomly computing the parameters is still guaranteed.

3.2. Integration in Networked Devices

A major improvement compared to the originally intended use of Uncoupled MAC working on dedicated embedded plug-in devices is the integration in networked devices. In this mode Uncoupled MAC can also run on one interface such as *eth*0 as shown in Fig. 4 enabling the application of the algorithm as a piece of software running on end devices or networking elements such as switches. This enables complete new fields of application for instance the utilization of Uncoupled MAC on ECUs or domain controllers in the automotive network domain.

In this mode of operation, a master and slave have to be defined at program start. The designated master device initiates the secure communication channel needed to pass status information and generation packets between the Uncoupled MAC partners. Instead of two hardware interfaces, only one interface is required and the internal ring-buffers for packet capturing are modified in a way that one captures only incoming traffic, while the other captures only outgoing traffic (Fig. 4). The separation is necessary in order to fulfill the same functionality of the architecture of embedded plug-in devices shown in [33] and [36] that obviate the bridge interface $br0$.

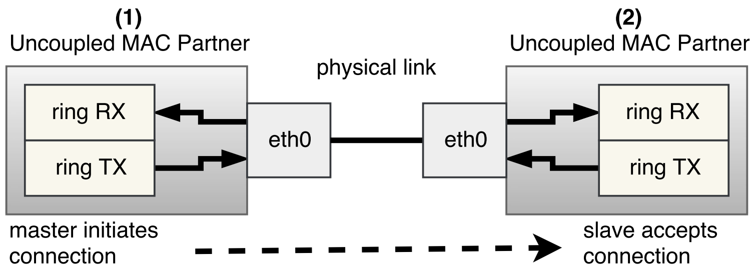

Fig. 4. Integration of Uncoupled MAC operation in networked devices

3.3. Synchronization of MAC Phase Start

A bottleneck of the decapsulation of MAC and message as with Uncoupled MAC is the start of the MAC Phases when making use of the proposed sampling technique (MAC Phase and Idle Phase). Similar to the so called *Two Generals Problem*, whose impossibility proof was first published in [38], the problem exists when two communicating Uncoupled MAC partners should start their MAC Phase. However, in the case of Uncoupled MAC different premises can be made by having a secure channel built by a certificate-based key establishment and an underlying time synchronization. By using a time synchronization protocol such as the Precision Time Protocol (PTP) to synchronize the master and slave, it is possible to facilitate accurate timestamps that can be used to synchronize the start of the MAC Phase for both participants. This synchronicity is crucial for guaranteeing a stable operation of the algorithm and minimizes the chance of packets being missed by one partner. Therefore, after the Idle Phase with a random duration α, a timestamp t_0 is taken by the master at the beginning of the Runtime Phase, to which an offset value t_{off} is added. The value t_{off} is at least a half of the Round Trip Time (RTT) between the master and slave ($\frac{1}{2} \times RTT$) and should be as small as possible, leaving only a minimal time window for an attacker to take advantage of. The resulting time value t_1 ($t_0 + t_{off}$) marks the start of the MAC Phase for both participants and is added to the regulation packet that is sent to the slave, thus allowing both partners to start the MAC Phase at the same time. The timing diagram in Fig. 5 depicts the timing synchronization mechanism.

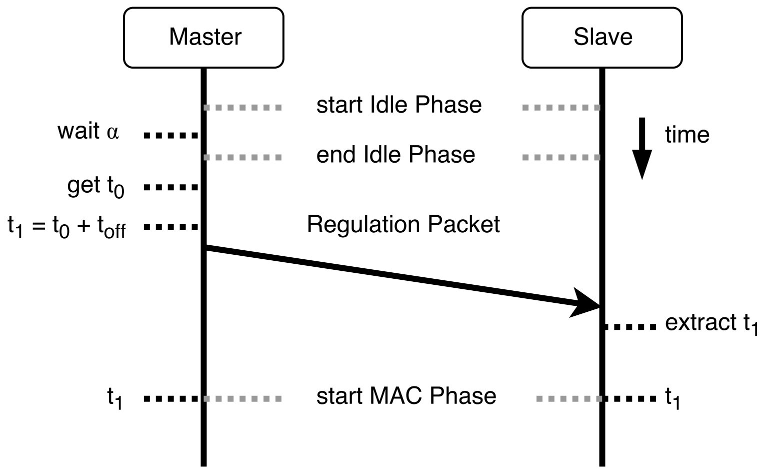

Fig. 5. Timing diagram for a synchronized start of a MAC Phase

3.4. Static Communication Mode

Depending on the network to secure, the embedded surveillance device presented in [36] may be omitted reducing the communication overhead towards each individual Uncoupled MAC partner. Especially in static networks, in which the number of Uncoupled MAC entities, communicating end-devices and thus the IP-addresses are defined, the static communication mode significantly speeds up the performance.

3.5. IDMEF Extension

The Intrusion Detection Message Exchange Format (IDMEF), an XML-based language, was specified as a RFC document to standardize the interfaces between intrusion detection, response and management components. Over the last decade several public and proprietary system-level standardization initiatives arose but they all ended in criticism, yielded weak adoption or have even been withdrawn as stated in [39] pursuing a standardized format. IDMEF has the advantages that the messages specified are well structured, the message fields (i.e. tags and attributes) have rigorous syntactic and semantic definitions and support better message expressivity [39].

A large number of popular IDSs (OSSEC, Barnyard2, Samhain, Surricata, ...) or SIEM-systems are using IDMEF. To ensure compatibility with those systems, Uncoupled MAC algorithm is extended by an IDMEF module. The proprietary defined info/error/warning message is replaced with a standardized format allowing a central authority to handle and manage security exceptions detected by Uncoupled MAC violations. Such violations have been discussed together with possible attack vectors and errors around the application of Uncoupled MAC in [36] and are listed in the following.

- HMAC packet inconsistencies (Hash mismatch, HMACs are not received, no HMAC for original packet, no feedback HMAC packet, ...)
- Sequence counter issues (Expected counter value is not within the tolerance window or counter value occurs twice, ...)
- Timestamp issues (Uncoupled MAC partners do not share the same time base, timestamp in HMAC packets exceed tolerance windows, ...)
- Authentication issues (Authenticated boot fails, direct attestation not possible, verifying certificates fails, ...)
- Connection issues (Direct, secure communication between Uncoupled MAC partners or surveillance device is terminated unexpectedly, a device does not report its active state, ...)

The support for IDMEF not only equips Uncoupled MAC with an IDS capability but also allows the interoperability with other IDSs allowing to find a consensus on alerts from various incident detection sources on a standardized format. This benefits the usage of alert analysis techniques not only on the devices themselves but also on a central more intelligent platform orchestrating different detection and reaction components as shown in Fig. 1 for systems that can, according to [40], be arranged distributed or decentralized.

4. Self-Regulation Algorithm

An Uncoupled MAC self-regulation algorithm is proposed in order to adjust the sampling parameters, the number of packets during a MAC Phase n and the duration of an Idle Phase α, on-the-fly. This allows for a given percentage q_0 of packets to be sampled/examined, which can be regarded as the adjustable security level, to increase or decrease the parameters n and α. This depends on the number of detected incidents z_D during a MAC Phase caused by Uncoupled MAC violations with respect to Section 3.5 and the number of unmeasured packets p_n during an Idle Phase.

Fig. 6 shows a schematic control circuit illustrating the self-regulation. The proposed mechanism might not only be applied for Uncoupled MAC but also for an IDS capable of sampling. This self-regulation approach can be utilized for a reasonable overhead and in the case of IDSs for a processable and manageable amount of data while still adapting to the systems' security state.

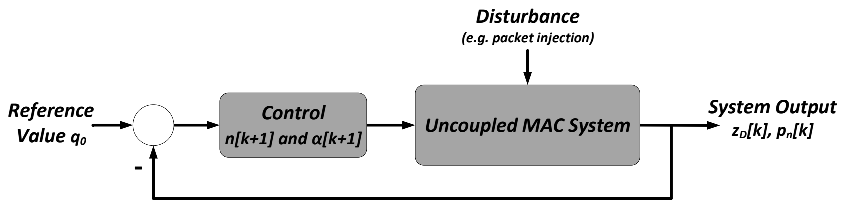

Fig. 6. Self-regulating sampling approach for Uncoupled MAC

According to the schematic from Fig. 6, formulas and algorithms for the Uncoupled MAC parameters n and α are derived to obtain self-regulation.

4.1. Number of Packets n per MAC Phase

In Eq. 1 the number of packets to be examined in the next MAC Phase $n[k+1]$ is calculated depending on the number of packets of the current MAC

Phase $n[k]$, the scaled number of packets identified by Uncoupled MAC detection ($z_D[k]$) and an intermediate value $q[k]$. This value $q[k]$ is responsible to keep the percentage of examined packets on a predefined value of q_0 depending on the number of unmeasured packets during the Idle Phase $p_n[k]$ (Eq. 2). The more packets are deemed as faulty by the algorithm, the higher the number of packets to be controlled in the next phase is set. However, if no Uncoupled MAC violations occur, the percentage of packets to be examined retains its value of q_0.

$$n[k+1] = n[k] + n_s \cdot z_D[k] + q[k] \tag{1}$$

$$q[k] = \left(q_0 - \frac{n[k]}{n[k] + p_n[k]} \right) \cdot n[k] \tag{2}$$

Where:

$n[k+1]$: packets to check in the next MAC Phase

$n[k]$: packets checked in the present phase

n_s: scale factor to set the impact of erroneous packets on the next phase

$z_D[k]$: number of erroneous packets detected in the current phase

$q[k]$: percentage of packets to check in the next phase

q_0: desired percentage of packets to check

$p_n[k]$: packets not checked due to the Idle Phase

4.2. Waiting Duration α in the Idle Phase

In Eq. 3 the waiting duration $\alpha[k+1]$ for the next Idle Phase is computed depending on a random value X_r within fixed boundaries $[\alpha_{MIN}, \alpha_{MAX}]$ and the scaled number of packets identified by Uncoupled MAC detection ($z_D[k]$) of the current MAC Phase.

$$\alpha[k+1] = X_r \in [\alpha_{MIN}, \alpha_{MAX}] - \alpha_s \cdot z_D[k] \tag{3}$$

Where:

$\alpha[k+1]$: duration of the next Idle Phase

X_r: random value in range $X_r = [\alpha_{MIN}, \alpha_{MAX}]$ calculated in each Idle Phase

α_s : scale factor to set the impact of erroneous packets on the next phase

$z_D[k]$: number of erroneous packets detected in the current phase

4.3. Formula Verification

A simple formula verification implemented in the *Python* programming language shows the desired behavior of $n[k+1]$ and $\alpha[k+1]$ depending on a predefined percentage of 60% of packets to be examined ($q_0 = 0.6$). The scaling factors in this example have been chosen to be $n_s = 0.3$ and $\alpha_s = 0.4s$. A total number of 1000 phases (MAC Phase + Idle Phase) have been examined. X_r is computed in each phase within the range of $[2s, 3s]$. Further, for each phase, a random number of unmeasured packets can be determined between the range $[3, 5]$. As shown in Fig. 7, two scenarios for detected packets (security incidents) by Uncoupled MAC ($z_D[k]$) are considered at $\frac{1}{3}$ and $\frac{2}{3}$ of the total of 1000 phases. At $\frac{1}{3}$, a constant number of 4 packets was detected for 100 phases resulting in a clear increase of $n[k+1]$ and a significant drop of $\alpha[k+1]$. However, after no more packets were detected, the values for n and α leveled off such that 60% of packets were examined again. At $\frac{2}{3}$, a steady increase of $z_D[k]$ shows clearly that more packets can be examined expressed by a steady increase of n and a shorter Idle Phase through a smaller α value. Thus, in a network having an increase on malicious activities more packets can be examined. It must be noted that Eq. 1 and 3 showing linear behavior regarding the term adjusted with z_D. However, in some cases, for instance when a faster adaption of n and α is desired, it might be necessary to consider non-linear behavior using a non-linear function $f(z_D)$.

4.4. Algorithm Notation

In order to implement Eq. 1 and 2 that dictate the behavior of the self-regulation system for the number of packets to be examined during a MAC

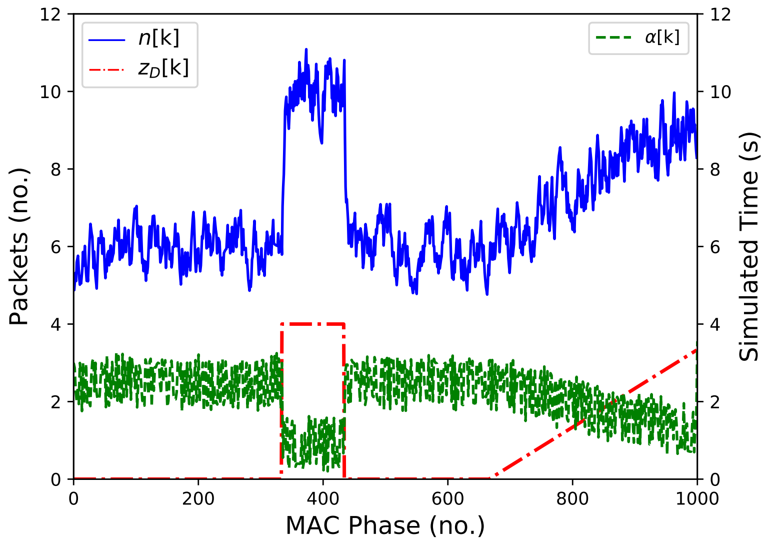

Fig. 7. Verification of the self-regulation formulas

Phase, Algorithm 1 is proposed. In addition to the computation given by the equations above, additional mechanisms must be added to the algorithm. One mechanism is to prevent $n[k+1]$ from falling below a threshold value N_{MIN}, ensuring that a minimum of packets is checked in each phase. Complementary to this, an upper threshold N_{MAX} must be defined to prevent $n[k+1]$ from reaching a too high value, producing excessive overhead.

For the α value described in Eq. 3, Algorithm 2 is proposed. In addition to the computation given by Eq. 3, two thresholds are added. These thresholds prevent α from reaching a value less than α_{MIN} or greater than α_{MAX}. This ensures that α is always a positive integer, not exceeding the specified Idle Phase duration value.

5. Evaluation

The proposed self-regulation algorithm and the Uncoupled MAC improvements have been added to the implementations presented in [33] and extended in [36]. The programming language is native *C* using the libraries *OpenSSL* for key establishment and HMAC-SHA-256 generation/verification in the MAC Phases,

Algorithm 1: Calculation of n for the next phase $k+1$

```
Input: $z_D[k]$, $n[k]$, $p_n[k]$
Output: $n[k+1]$
    Initialization : $q[k]$
    Constants : $N_{MIN}$, $N_{MAX}$, $q_0$, $z_s$
 1: $q[k] \leftarrow q_0 - (n[k]/(n[k] + p_n[k])) \cdot n[k]$
 2: if  $(q[k] < 0)$  then
 3:     $q[k] \leftarrow 0$
 4: end if
 5: $n[k+1] \leftarrow n[k] + z_s \cdot z_D[k] + q[k]$
 6: if  $(n[k+1] < N_{MIN})$  then
 7:     $n[k+1] \leftarrow N_{MIN}$
 8: else if  $(n[k+1] > N_{MAX})$  then
 9:     $n[k+1] \leftarrow N_{MAX}$
10: end if
11: return  $n[k+1]$
```

PF_RING in combination with *libpcap* for packet processing and *libprelude* for IDMEF support. The evaluation in this article deals with customized attack scenarios in an environment targeted to exploit the weaknesses of Uncoupled MAC's concept of Idle Phases to show their impact on the detection capability by also examining the trade-off of preserving resources. It is therefore no aim to compare the IDS functionality with other attack detection mechanisms in this work since Uncoupled MAC is a cryptographic scheme by nature and provides as a side effect simple IDS functionality benefiting environments characterized by resource constraints and static communication. Especially a large network data diversity, not provided with the evaluation environment, is a mandatory aspect to apply anomaly-based machine learning mechanisms in order to properly learn the network behavior.

Algorithm 2: Calculation of α for the next phase $k+1$

```
Input: $X_r$, $z_D[k]$
Output: $\alpha$
    Constants : $\alpha_s$, $\alpha_{MAX}$, $\alpha_{MIN}$
1: $\alpha \leftarrow X_r - \alpha_s \cdot z_D[k]$
2: if $(\alpha < \alpha_{MIN})$ then
3:     $\alpha \leftarrow \alpha_{MIN}$
4: end if
5: if $(\alpha > \alpha_{MAX})$ then
6:     $\alpha \leftarrow \alpha_{MAX}$
7: end if
8: return $\alpha$
```

5.1. Virtualized Environment

A virtualized environment is preferred compared to the Simulation Assessment Model presented in [29] since the simulation is an idealized model of the algorithm with many constraints such as the neglection of bidirectional communication. The virtualized environment behaves like an authentic evaluation with real (embedded) devices since among others the hardware and network resources and constraints can be evaluated as well. For more flexibility in carrying out exhausting measurements evaluating various scenarios for trend estimation of Uncoupled MAC behavior, the self-regulation algorithm might be implemented as an extension for the Simulation Assessment Model in future work.

The virtualized evaluation environment is implemented on a virtual *Proxmox* platform for scalability and flexibility. All machines as well as the intermediary devices are virtual entities on the same virtualization host as illustrated in Fig. 8 including the associated hardware constraints.

In order to evaluate both, the plug-in and integrated mode, two Uncoupled MAC Devices and two Legacy Devices are used. Testing in the integrated device mode having only communication between the Uncoupled MAC Devices each evaluation ends up with the same results compared to the plug-in mode in which the Uncoupled MAC Devices secure the Legacy Device communication. To connect the Legacy Devices with the Uncoupled MAC Devices, a simple virtual Proxmox-internal bridge is sufficient. The Uncoupled MAC Devices are connected via an Open vSwitch (OVS) to allow port-mirroring and packet injection by the attacker, which is connected to the same switch.

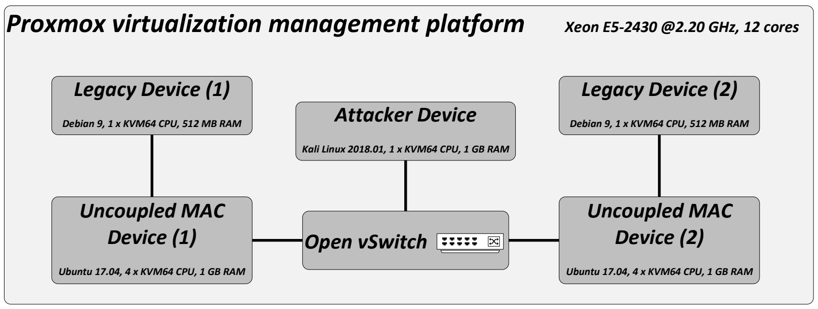
Proxmox virtualization management platform
Xeon E5-2430 @2.20 GHz, 12 cores
Legacy Device (1)
Debian 9, 1 x KVM64 CPU, 512 MB RAM
Attacker Device
Kali Linux 2018.01, 1 x KVM64 CPU, 1 GB RAM
Legacy Device (2)
Debian 9, 1 x KVM64 CPU, 512 MB RAM
Uncoupled MAC Device (1)
Ubuntu 17.04, 4 x KVM64 CPU, 1 GB RAM
Open vSwitch
Uncoupled MAC Device (2)
Ubuntu 17.04, 4 x KVM64 CPU, 1 GB RAM

Fig. 8. Structure of the virtual evaluation environment

In the following evaluations for plug-in and integrated mode, the traffic between the Legacy Device (1) and (2) is secured by the Uncoupled MAC Devices (1) and (2) and originates from Legacy Device (1). For evaluation purposes, the Internet Control Message Protocol (ICMP) is used in form of ICMP echo-requests and corresponding replies. An Attacker Device is impersonating Legacy Device (1) and is injecting packets to test the Uncoupled MAC detection capability in different scenarios. In addition to this, several conditions and test parameters are defined:

1. the interval between ICMP echo-requests is set to 500 ms simulating a periodic communication
2. the standard size for an ICMP echo-request and echo-reply is used (98 bytes total length, 48 bytes payload)
3. a maximum of 32 packets for n is checked per MAC Phase
4. a minimum of 8 packets for n is checked per MAC Phase
5. a maximum value of 50 (5 s) for α is determining the Idle Phase maximum

for the attack scenarios continuous and stochastic, while α is set to 42 (4.2 s) as an Idle Phase maximum value for the single and burst injection attacks; the details on each attack scenario are described in Section 5.2

6. a minimum value of 3 (300 ms) for α is determining the Idle Phase minimum for all attack scenarios
7. Uncoupled MAC Device (1) is set as the master, Uncoupled MAC Device (2) as the slave
8. the master device is responsible for HMAC verification, the slave device is responsible for HMAC generation
9. the Uncoupled MAC Devices are synchronized via the PTP protocol, enabling precise start times for the MAC Phases on both devices and accurate generation packet timestamps

5.2. Attack Scenarios

Focus of the following customized attack scenarios is to exploit the Uncoupled MAC algorithm's Idle Phase times in order to inject spoofed data. In those phases no packet authenticity and integrity checks are performed. Since any other attack type (modification, replay or man-in-the-middle) as discussed in [36] would also be detected due to Uncoupled MAC violations (Section 3.5), only variations of packet injection attacks are defined to simulate different adversary skill levels. In order to evaluate the reliability of detection and the benefits of the self-regulated version of Uncoupled MAC compared to the non-self-regulated one, the following scenarios are specified:

1. *Continuous Injection* - a single packet is injected in fixed intervals
2. *Stochastic Injection* - a random number of packets in a specific range is injected at random intervals with a fixed minimum and maximum time between injections
3. *Bandwidth Low Injection* - one (single) or multiple (burst) packets are injected after a bandwidth low is detected indicating an Idle Phase
4. *Weak Spot Injection* - a single packet is injected immediately after the Regulation Packet is sent out by the master and detected by the attacker

For single and burst bandwidth low injection an intelligent adversary might monitor the cycle of MAC Phases and Idle Phases and learns the average bandwidth overhead in order to inject packets after a bandwidth low is detected indicating an Idle Phase. One could say that an attacker who would be able to only monitor the actual network traffic produced by Uncoupled MAC, e.g. by monitoring Uncoupled MAC's port number, would be able to inject packets without measuring the bandwidth. He would though be able to see when no generation and verification packets are exchanged indicating an Idle Phase (if no heartbeat messages are applied) but could not determine an actual trigger value because of Uncoupled MAC's random parameters n and α. Thus, a learning phase by a bandwidth measurement over multiple phases is necessary in order to gain a dedicated trigger value for injection. It must be noted that these scenarios are rather theoretical since in a realistic network environment more than two parties communicate resulting in an even higher bandwidth utilization with fluctuations such that it might not be possible to determine the actual bandwidth overhead by Uncoupled MAC in order to detect the Idle Phases through bandwidth lows.

Weak spot injection assumes that an attacker is able to identify the regulation packet on the network data. However, as stated in [36], Uncoupled MAC communication is transferred over a secure encrypted channel with applied heartbeat messages. Thus, on the one side the communication overhead increases but leaves no possibility that an attacker is able to distinguish a regulation packet from a heartbeat message. Since these features are complementary depending on the desired security level and possible overhead, the evaluation deals with a lightweight version not considering applied heartbeat messages.

5.3. Attacker Implementation

The attack scenarios are implemented using Python-based scripts. For continuous injection, the script is used to inject packets at an interval of two seconds between injections. The stochastic injection sends a random count of attacker packets $a_{packets} = [a_{min}, a_{max}]$ at random times $\Delta t_{random} = [\Delta t_{min}, \Delta t_{max}]$.

Both scenarios could also represent a non-malicious behavior for instance when a network participant acts as a so called "babbling idiot" in sending out continuous or stochastic packets as a result of a malfunction. The latency of the attacker script must also be considered in the evaluation of the continuous and stochastic attacks. If, for example, the script is started during a MAC Phase, the first injection may take place in the Idle Phase, resulting in a low detection rate for the first phase. On the other hand, if the script is started in an Idle Phase, a high detection rate might be achieved for the first phase. This non-deterministic behavior of the attacker program results from first constructing the attacker packet at the start of the script and the delay introduced by the sockets.

For the bandwidth low injection, an intelligent attacker program is developed. This intelligent attacker program first sniffs 120 packets passed between the two Uncoupled MAC partners in a learning phase. During this phase, bandwidth data are computed and stored, which is then assessed in the processing phase. This phase yields mean values for the upper, lower and middle bandwidth. The mean values can then be used in the attack phase, in which the bandwidth is continuously measured using a sampling rate of 10 ms. Then packets are injected either in single (a single packet) or burst mode (e.g. three packets) each time a bandwidth low is reached. The bandwidth low marks the start of an Idle Phase for the attacker, thus leaving a confined time window for an attack. An example bandwidth measurement of the attacker in an arbitrary attack phase with injected packets when a bandwidth low is detected is shown in Fig. 9.

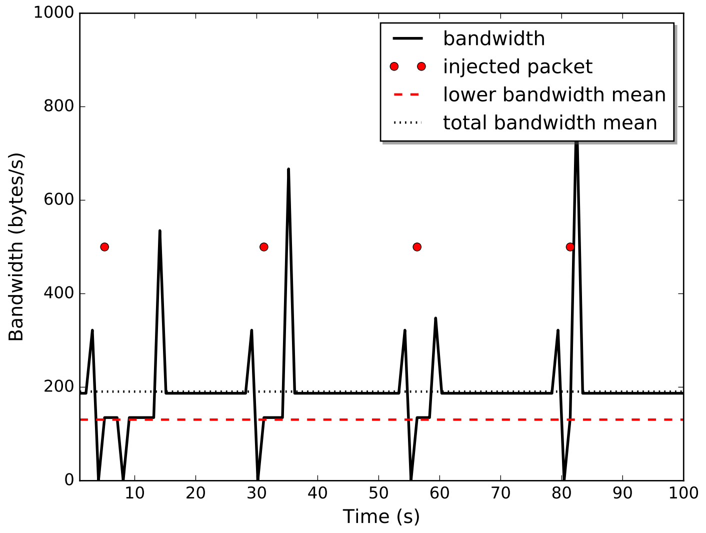

Fig. 9. Attacker bandwidth measurement and injected packets in detected bandwidth lows

5.4. Evaluation Metrics

For the evaluation of IDSs, especially for the problem of statistical classification, different characteristic values are used. One of the most important notations is to use the parameters derived from the Confusion Matrix (Table 1) as stated among other literature in [41]. This specific table allows the representation of the performance of an algorithm, typically used for machine learning but in this context used to build a bridge towards Uncoupled MAC's incident detection functionality.

Table 1: Confusion Matrix for IDS evaluation

	Actual Non-Anomaly	**Actual Anomaly**
Predicted Non-Anomaly	true negative (TN)	false negative (FN)
Predicted Anomaly	false positive (FP)	true positive (TP)

Where:

TP: intrusion/anomaly classified as an intrusion/anomaly

FP: normal event/behavior classified as an intrusion/anomaly

TN: normal event/behavior classified as a normal event/behavior

FN: intrusion/anomaly classified as a normal event/behavior

From the parameters of Table 1, formulas for the computation of many other characteristic values (sensitivity, specificity, accuracy, etc.) can be derived. For Uncoupled MAC, the following assumptions can be made: Due to the nature of MAC generation and verification in a MAC Phase, no false positives can be obtained (FP=0) since modified or injected packets can certainly be detected to be true positives. Unexamined packets within the Idle Phase will either yield true negatives in the case of non-malicious packets or false negatives in the other case. The possible conditions for Uncoupled MAC in an arbitrary phase are shown in the example of Fig. 10.

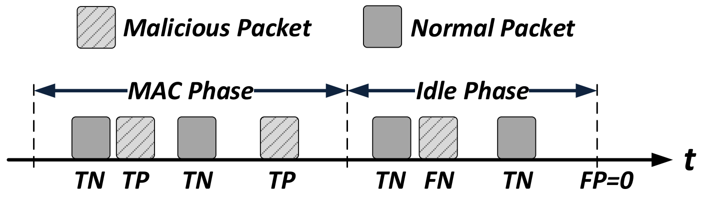

Fig. 10. Confusion Matrix parameters for some Uncoupled MAC phase k

The quantity used to measure the efficiency of an algorithm's self-regulation and non-self-regulation is the detection rate since it is according to [25] the most important metric for an IDS. For Uncoupled MAC evaluation it is defined as the percentual value of the number of injected attacker packets versus the number of these packets detected by the Uncoupled MAC algorithm. The detection rate (DR) described in percent corresponds to the true positive rate (TPR) or sensitivity derived from the Confusion Matrix which can be computed according to Eq. 4. In the example from Fig. 10, for some phase k, $TP = z_D[k]$ equals 2 and $FN = 1$ yields a detection rate $DR = TPR = \frac{2}{2+1} = 67\%$.

$$TPR = \frac{TP}{TP + FN} \tag{4}$$

6. Measurement Results

The results from measurements, taken within the evaluation environments while multiple attack scenarios are executed, are described in the following sections. The values for non-self-regulation are set such that comparable to self-regulation approximately $(q_0 \times 100)\%$ of packets are examined by default.

6.1. Continuous Injection

As described in Section 5.2, a continuous attack is carried out on both the self-regulation and the non-self-regulation algorithm. In this attack mode, a single attacker packet z_D is injected into the network in fixed intervals of two seconds. The scenario has been carried out ten times for each algorithm over a total of 30 MAC phases. The mean values for the detection rates are calculated for both the non-self-regulation algorithm and the self-regulation algorithm with the following set of parameters for self-regulation: $n_s = 1.30$, $\alpha_s = 2.0$, $q_0 = 0.75$.

While the non-self-regulation version of Uncoupled MAC performs poorly in this scenario, the self-regulation version produces a significantly higher mean detection rate (Fig. 11). By adjusting the Uncoupled MAC parameters n and α, the detection rate increases over runtime due to the increase of n and average decrease of α following an exponential approximation curve for self-regulation. However, both algorithms would reliably detect an attacker or a malfunctioning device acting as a "babbling idiot".

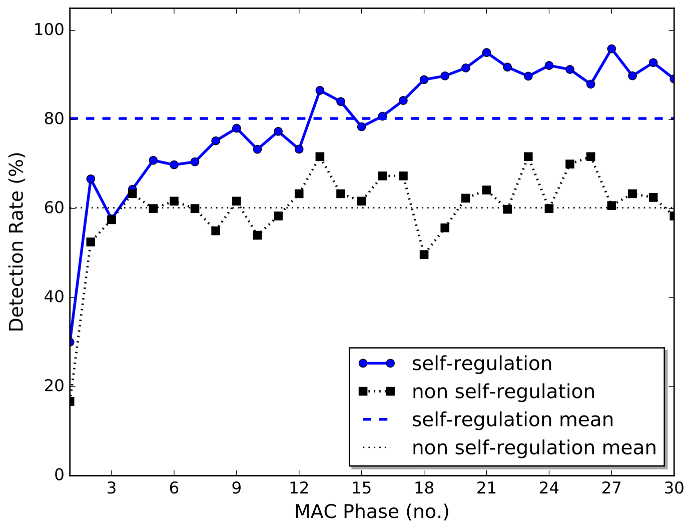

Fig. 11. Detection rate - continuous packet injection

6.2. Stochastic Injection

In this attack scenario the attacker sets a minimum and a maximum of packets z_i to inject. For this test, the interval $z_i \in [1;4]$ was chosen. Further, a minimum and maximum for the time between injections t_i (in seconds) needs to be fixed, which in this case is set to the interval $t_i \in [0.5;10]$. A number of packets to inject (z_i) and a wait time (t_i) are selected randomly from the respective interval. The attack is carried out 10 times over 30 phases with the following parameters chosen for the self-regulation algorithm: $n_s = 0.80$, $\alpha_s = 2.0$, $q_0 = 0.75$.

The mean values for the detection rates are calculated for both the non-self-regulation algorithm and the self-regulation algorithm and illustrated in Fig. 12. Compared to the continuous injection, the average detection rate is smaller due to a greater difficulty to detect random injection. Further, the fluctuation expressed by the curve is greater because of the mean computation of multiple stochastic measurements. However, again the self-regulation performs better than non-self-regulation.

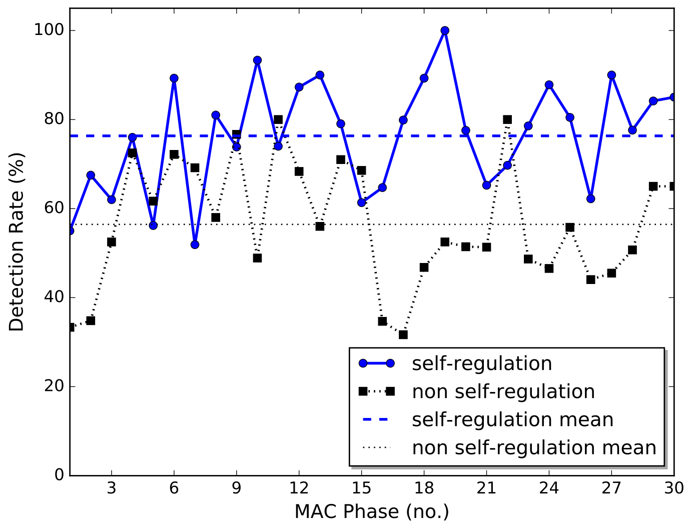

Fig. 12. Detection rate - stochastic attack mode

6.3. Bandwidth Low Injection

A more sophisticated attack scenario compared to the continuous or stochastic injection is the exploitation of the Idle Phase by measuring the typical MAC Phase and Idle Phase period and injecting packets when a bandwidth low representing an Idle Phase is detected. Two subscenarios are performed. One in which only a single packet is injected by an attacker and a second in which more information is transferred in form of multiple injected packets.

Single Injection

Fig. 13 depicts the results of an attack carried out over a time period of 300 seconds for both variants of the Uncoupled MAC algorithm. The chart shows the duty-cycle of Uncoupled MAC, alternating between MAC and Idle Phases. The packets that are detected by Uncoupled MAC are shown in green, the packets that were not detected are shown in red. While the non-self-regulation algorithm does not exceed a detection rate of 20%, the self-regulation algorithm detects about 50% of the attacker packets on average. When an attacker packet is detected by the self-regulation algorithm, the duration of the Idle Phase (α) is reduced and the number of packets to be checked is increased (n) as described in Section 4. Due to this behavior, the margin for packet injection is also reduced, resulting in fewer possibilities for an attack, while the detection rate of Uncoupled MAC increases. For this scenario the following parameters where chosen for the self-regulation algorithm: $n_s = 18.0$, $\alpha_s = 25.0$, $q_0 = 0.75$. By choosing stricter parameters, the impact of a single attacker packet on the resulting behavior is greater than with the scenarios described in Section 6.1 and 6.2, resulting in a faster reaction compared to a detected injection in the continuous and stochastic attack scenarios.

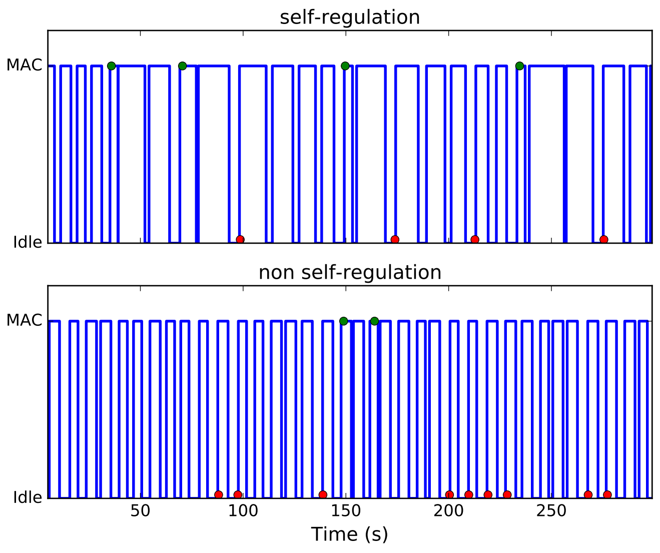

Fig. 13. Packet detection per phase - single attack mode

Burst Injection

Fig. 14 depicts the results of an attack carried out over a time period of 300 seconds for both variants of the Uncoupled MAC algorithm. The chart shows again the duty-cycle of Uncoupled MAC, alternating between MAC and Idle Phases when three packets are injected per bandwidth low. The detected packets are shown in green, the undetected packets in red. Again the margin for the attacker program is reduced by the self-regulation algorithm. In this scenario the number of attacker packets introduced to the test network is greater than with the single injection scenario, making the limitation of the attacker scope more distinct. While 72 packets are injected with the non-self-regulation algorithm having a detection rate of approximately 47%, only 21 packets are injected for the self-regulation variant (detection rate of approximately 62%), due to the fact that the self-regulation variant dynamically adjusts its behavior with respect to the attacker packets detected. In this scenario, the same parameters as with the single injection were selected for the self-regulation algorithm, again with the aim to maximize the detection rate after an attack was registered as fast as possible.

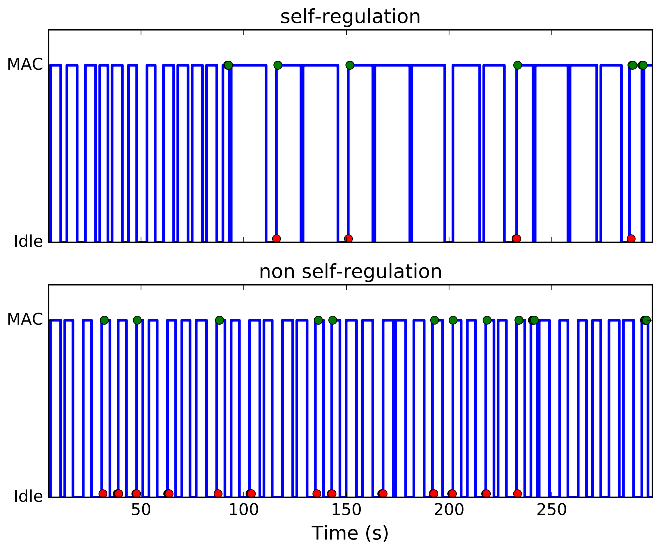

Fig. 14. Packet detection per phase - burst attack mode

6.4. Weak Spot Injection

As described in Section 3.3, the ideal offset t_{off} to start a new MAC Phase initiated by the master in sending a Regulation Packet to the slave is $\frac{1}{2} \times RTT$. However, in a practical implementation when referring to Fig. 5, t_{off} must include deviations for instance due to network jitter, processing delays for transmitting and receiving the Regulation Packet or delays considering the timestamp and t_{off} generation. Therefore it must be typically larger than $\frac{1}{2} \times RTT$. Even if according to the concept the exchanged Uncoupled MAC information is encrypted and thus an attacker cannot identify the Regulation Packet directly, in this evaluation he might be able to exploit the small time window after an Idle Phase ends until a new synchronized MAC Phase starts (weak spot) to inject a malicious packet.

The injection of a single packet of the intelligent attacker described in Section 5.2 is triggered immediately after identifying a Regulation Packet for a total number of 15 phases. Due to the fact that the t_{off} value is implemented in both algorithms with and without self-regulation in an identical way, and independent of the calculation of n and α values, the stated facts concern both algorithms likewise. The detection rate of Uncoupled MAC for changing integer multiples of the RTT is illustrated in Fig. 15 for Uncoupled MAC with applied self-regulation. The average RTT in this scenario derived utilizing the *ping* tool is 0.804 ms. The greater the value for t_{off}, the less malicious packets are detected. For a practical implementation of Uncoupled MAC, thus, values for $t_{off} \in [1; 5[\times RTT$ are recommended for this scenario in order to mitigate weak spot injection. However, even a large value for t_{off} of approximately $75 \times RTT$ would still detect attacker packets under the set conditions. The graph states further the estimated detection rate in network environments in which the Regulation Packet is delayed for instance due to overloaded network switches up to a maximum of $100 \times RTT$ when applying self-regulated Uncoupled MAC.

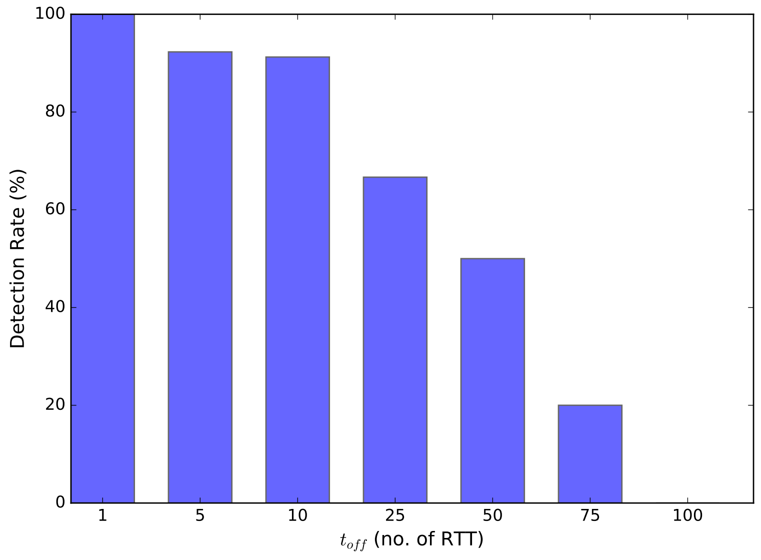

Fig. 15. Detection rate - weak spot injection

6.5. Uncoupled MAC Overhead

In this section two of the aforementioned disadvantages of Uncoupled MAC are addressed. On the one side the additional network utilization is measured including the presence of an attacker and on the other side the overhead on resources based on a CPU and memory measurement required to perform the Uncoupled MAC algorithm.

Network Utilization

Uncoupled MAC and especially self-regulation increases the overhead of network utilization in presence of an attacker since its parameters are dynamically adjusted due to the detection of the mechanisms' violations. In order to show the impact of different attacker behavior on the network overhead, the continuous injection was modified such that the interval between each single packet injection is decreased over runtime simulating an increasing attacker load. Fig. 16 is illustrating this scenario for an Uncoupled MAC setting of $n_s = 1.30$, $\alpha_s = 2.0$, $q_0 = 0.75$, an underlying basic ICMP traffic with a total packet size of 1008 bytes and an attacker that injects packets after approximately 20 seconds. The range of injection is $[2, 0.25]$ in seconds having a decreasing interval of 0.05. The attack is carried out 10 times showing the averaged curves in Fig. 16. Until approximately 20 seconds no attacker packets are injected resulting in an overhead of 8%. After approximately 20 seconds the injected attacker packets lead to a steady increase of the total ICMP bandwidth. However, even if Uncoupled MAC detects the injected attacker packets and adjust its n and α values, the overhead stays about the same with a slight increase to 9%. The reason for this are the random MAC and Idle Phases that interfere for the averaged 10 attack measurements. In a real network environment the overhead by Uncoupled MAC including self-regulation would therefore remain nearly constant when considering the mean bandwidth utilization.

It must be noted that the overhead of Uncoupled MAC, even when an attacker is present, depends on two factors. One impact is the security configuration of Uncoupled MAC for instance whether using additional feedback messages or not which would increase or decrease the overhead. For the network utilization measurement no feedback messages are considered. On the other side the packet size of the underlying protocol to be secured plays a major role. Independent of this size, the size of Uncoupled MAC generation packets is fixed, mainly characterized by the 256-bit HMAC. The percentual Uncoupled MAC overhead would therefore increase for a minor basic packet size.

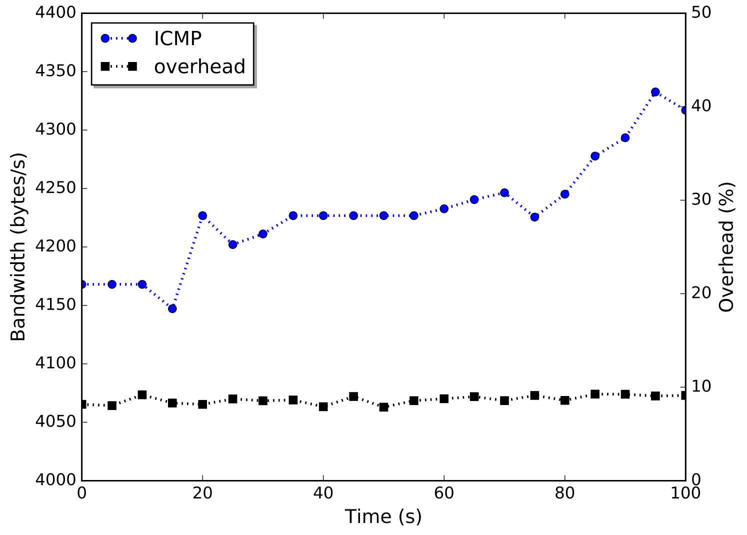

Fig. 16. Bandwidth overhead in presence of an attacker

CPU and Memory Utilization

IDSs, especially complex machine learning based anomaly detection algorithms, demand expensive resources [42]. Due to the nature of Uncoupled MAC, a certain resource consumption overhead is present when verifying the integrity and authenticity of a specified q_0 percent of packets. Fig. 17 shows the CPU (left plot) and memory (right plot) utilization of the self-regulated Uncoupled MAC process with $q_0 = 0.75$ carried out on the Uncoupled MAC Device (1) applying the *psrecord* and the *Massif (Valgrind)* tool. In particular, the memory utilization is split into the heap and stack utilization.

The figure covers the Setup Phase with the key agreement, one Idle Phase and one MAC Phase in generation mode for the Uncoupled MAC master. The Setup Phase in the current implementation is quite CPU consuming but needs to be done only once for each Uncoupled MAC connection setup ($\approx$ 0-2 s). This phase also results in fluctations of the heap and stack memory. After the Setup Phase, a thread is created performing the HMAC generation and verification which results in a constant heap and stack memory utilization of approximately

8.45 kB (heap) and 5.14 kB (stack) for all subsequent Idle and MAC Phases. The Idle Phases do not demand any CPU load since the master's process suspends for a random time depending on the value of α ($\approx$ 2-10.5 s). In the experiment after approximately 10.5 seconds the MAC Phase starts showing minor peaks for each computed HMAC. The CPU overhead produced by Uncoupled MAC even during MAC Phases is, however, decent with only approximately 10% on average.

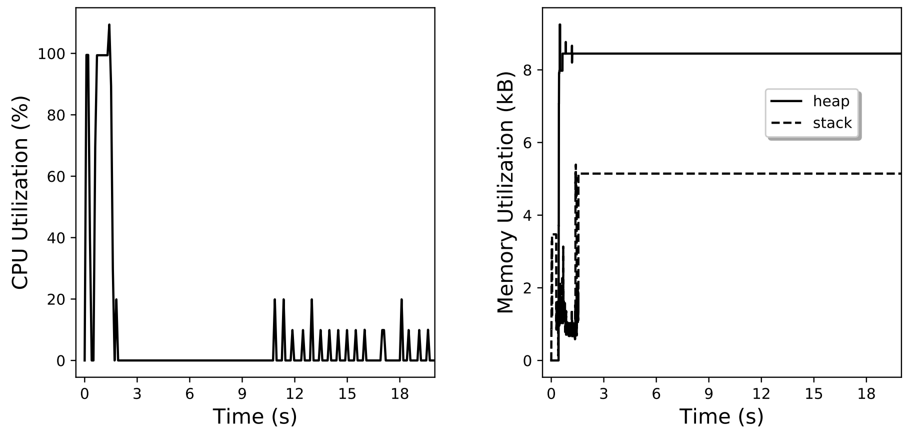

Fig. 17. CPU (left) and memory (right) utilization of the Uncoupled MAC process

7. Conclusions and Future Work

In this work Uncoupled MAC has been presented as a protection-based security technique improved and extended to work as a detection-based method as well. By the sampling approach having MAC Phases and Idle Phases, a certain security level can be adjusted with a balanced overhead on resource consumption and network utilization. The proposed self-regulation mode examines a specifiable amount of packets in network environments with security incidents happening over the average runtime. However, the more malicious activities are detected, the more active Uncoupled MAC becomes showing behavior of a partial working IDS component. Applying self-regulation, malicious actions can be detected quite fast and reliably compared to classical Uncoupled MAC.

Securing layer-2 protocols is either complex to implement or popular IDS solutions do not cover data link layer detection. Having a layer-2 support and the functionality to work integrated in devices, Uncoupled MAC enables completely new fields of applications in a holistic approach. As a combination of a protection- and detection-based method it can be applied as retrofitted security add-on in dedicated embedded plug-in devices or as a software application running transparently in the background on end-devices or network elements while utilizing tolerable resource consumption over the average runtime.

The concept of self-regulation shall be extended to other IDS systems performing sampling in future research work and a more intelligent interaction, for instance by applying alert analysis techniques. The basis by extending Uncoupled MAC with IDMEF is given for this. An even more intelligent sampling mechanism based on a combination of statistics, local, global and feedback from an IDS cluster as proposed in [25] could be added to the self-regulation in order to further improve the interworking of detection and protection approaches with resource conservation.

Since Uncoupled MAC, as a cryptographic scheme by nature, is only a possibility for incident detection providing basic IDS functionality, it might be of interest to compare it with other attack detection mechanisms by examining the trade-off between detection capability and resource consumption. Part of further work is therefore to set up an appropriate evaluation environment, e.g. [43, 44], with, among others, a larger protocol and network data diversity in order to properly train the models for anomaly-based machine learning algorithms, for instance Loda [45], and integrating known attack scenarios such that also misuse-based IDS can be compared.

Especially targeted towards future-orientated IoT-enabled applications, Uncoupled MAC shall be implemented in an even more lightweight variant applying for instance the lightweight MAC Chaskey [46] or LMAC [47] for MAC generation and verification together with the MQTT protocol for lightweight Uncoupled MAC partner communication. By using a more lightweight MAC function, the additional overhead not only from a CPU but also from a memory point of view can be reduced. From the bandwidth utilization perspective, using for instance LMAC, can decrease the load by using the 64-bit digest instead of the 256-bit HMAC mainly determining the generation's message size.

Acknowledgements

This research work has been supported by the research projects no. 16KIS0539 and 13FH645IB6 of the German Federal Ministry of Education and Research and by the Ministry of Education, Youth and Sports of the Czech Republic under grant No. LO1506.

References

[1] S. Popov, IOTA whitepaper, Tech. rep., IOTA Foundation (2017).
URL `https://iota.org/IOTA_Whitepaper.pdf`

[2] R. Srinivasan, A. Sharmili, S. Saravanan, D. Jayaprakash, Smart vehicles with everything, in: 2016 2nd International Conference on Contemporary Computing and Informatics (IC3I), 2016, pp. 400–403. `doi:10.1109/IC3I.2016.7917997`.

[3] T. R. Vittor, T. Sukumara, S. D. Sudarsan, J. Starck, Cyber security - security strategy for distribution management system and security architecture considerations, in: 2017 70th Annual Conference for Protective Relay Engineers (CPRE), 2017, pp. 1–6. `doi:10.1109/CPRE.2017.8090020`.

[4] I. Studnia, E. Alata, V. Nicomette, M. Kaâniche, Y. Laarouchi, A language-based intrusion detection approach for automotive embedded networks, in: The 21st IEEE Pacific Rim International Symposium on Dependable Computing (PRDC 2015), Zhangjiajie, China, 2014.
URL `https://hal.archives-ouvertes.fr/hal-01419020`

[5] B. Arrington, L. Barnett, R. Rufus, A. Esterline, Behavioral modeling intrusion detection system (bmids) using internet of things (iot) behavior-based anomaly detection via immunity-inspired algorithms, in: 2016 25th

International Conference on Computer Communication and Networks (ICCCN), 2016, pp. 1–6. doi:10.1109/ICCCN.2016.7568495.

[6] M. Heigl, M. Schramm, D. Fiala, A lightweight quantum-safe security concept for wireless sensor network communication, in: 2019 IEEE International Conference on Pervasive Computing and Communications Workshops (PerCom Workshops), 2019.

[7] E. Cayirci, C. Rong, Security in wireless ad hoc and sensor networks, in: Chichester, West Sussex, United Kingdom,: John Wiley & Sons, Ltd.), 2009. doi:10.1002/9780470516782.

[8] K.-T. Cho, K. G. Shin, Fingerprinting electronic control units for vehicle intrusion detection, in: 25th USENIX Security Symposium (USENIX Security 16), USENIX Association, Austin, TX, 2016, pp. 911–927. URL https://www.usenix.org/conference/usenixsecurity16/technical-sessions/presentation/cho

[9] K. Scarfone, P. Mell, Guide to intrusion detection and prevention systems (idps), Tech. rep., National Institute of Standards & Technology (2007). URL https://www.nist.gov/publications/guide-intrusion-detection-and-prevention-systems-idps

[10] B. Caswell, J. Beale, A. Baker, Snort Intrusion Detection and Prevention Toolkit, 1st Edition, Syngress, 2007.

[11] F. Yulong, Y. Zheng, C. Jin, K. Ousmane, C. Xuefei, An automata based intrusion detection method for internet of things, Mobile Information Systems 2017 (2017) 118–173. doi:10.1155/2017/1750637.

[12] A. N. Jaber, M. F. Zolkipli, H. A. Shakir, M. R. Jassim, Host based intrusion detection and prevention model against ddos attack in cloud computing, in: Advances on P2P, Parallel, Grid, Cloud and Internet Computing, Springer International Publishing, Cham, 2018, pp. 241–252.

[13] T. Mahbod, An adaptive hybrid intrusion detection system, Ph.D. thesis, University of New Brunswick (2011).
URL `http://www.academia.edu/1123827/An_Adaptive_Hybrid_Intrusion_Detection_System`

[14] D. Fallstrand, V. Lindstroem, Applicability analysis of intrusion detection and prevention in automotive systems, Master's thesis, Chalmers University of Technology Goteborg (2015).
URL `http://publications.lib.chalmers.se/records/fulltext/219075/219075.pdf`

[15] R. A. R. Ashfaq, X.-Z. Wang, J. Z. Huang, H. Abbas, Y.-L. He, Fuzziness based semi-supervised learning approach for intrusion detection system, Information Sciences 378 (2017) 484 – 497. `doi:https://doi.org/10.1016/j.ins.2016.04.019`.
URL `http://www.sciencedirect.com/science/article/pii/S0020025516302547`

[16] C. Kruegel, F. Valeur, G. Vigna, Intrusion Detection and Correlation, Springer, Boston, MA, 2005. `doi:10.1007/b101493`.

[17] A. Taylor, Anomaly-based detection of malicious activity in in-vehicle networks, Ph.D. thesis, University of Ottawa (2017).
URL `https://ruor.uottawa.ca/bitstream/10393/36120/3/Taylor_Adrian_2017_thesis.pdf`

[18] G. Androulidakis, S. Papavassiliou, Improving network anomaly detection via selective flow-based sampling, IET Communications 2 (3) (2008) 399–409. `doi:10.1049/iet-com:20070231`.

[19] J. Mai, A. Sridharan, C. N. Chuah, H. Zang, T. Ye, Impact of packet sampling on portscan detection, IEEE Journal on Selected Areas in Communications 24 (12) (2006) 2285–2298. `doi:10.1109/JSAC.2006.884027`.

[20] E. G. Bakhoum, Intrusion detection model based on selective packet sampling, EURASIP Journal on Information Security 2011. doi:10.1186/1687-417X-2011-2.
URL https://doi.org/10.1186/1687-417X-2011-2

[21] T. Ha, S. Kim, N. An, J. Narantuya, C. Jeong, J. Kim, H. Lim, Suspicious traffic sampling for intrusion detection in software-defined networks, Computer Networks 109 (2016) 172 – 182, traffic and Performance in the Big Data Era. doi:https://doi.org/10.1016/j.comnet.2016.05.019.
URL http://www.sciencedirect.com/science/article/pii/S1389128616301645

[22] R. E. Jurga, M. M. Hulboj, Technical report - packet sampling for network monitoring, Tech. rep., CERN openlab report (2007).
URL http://openlab.cern/publications/technical_documents/packet-sampling-network-monitoring

[23] L.-B. Xu, G.-X. Wu, J.-F. Li, Packet-level adaptive sampling on multi-fluctuation scale traffic, in: Proceedings. 2005 International Conference on Communications, Circuits and Systems, 2005., Vol. 1, 2005, pp. 604–608 Vol. 1. doi:10.1109/ICCCAS.2005.1493481.

[24] K. Bartos, M. Rehak, V. Krmicek, Optimizing flow sampling for network anomaly detection, in: 2011 7th International Wireless Communications and Mobile Computing Conference, 2011, pp. 1304–1309. doi:10.1109/IWCMC.2011.5982728.

[25] Q. Xia, T. Chen, W. Xu, Cids: Adapting legacy intrusion detection systems to the cloud with hybrid sampling, in: 2016 IEEE International Conference on Computer and Information Technology (CIT), 2016, pp. 508–515. doi:10.1109/CIT.2016.31.

[26] J. He, Y. Yang, X. Wang, Z. Tan, Adaptive traffic sampling for p2p botnet detection, International Journal of Network Management 27 (5) (2017)

e1992–n/a, e1992 nem.1992. doi:10.1002/nem.1992.
URL http://dx.doi.org/10.1002/nem.1992

[27] B. S. Shankar, A. T. Murthy, The detailed study and verities of macsec in wired ethernet, in: International Journal of Emerging Trends in Engineering and Development, Vol. 4, 2015, pp. 508–512.

[28] A. Kushwaha, H. R. Sharma, A. Ambhaikar, A novel selective encryption method for securing text over mobile ad hoc network, Procedia Computer Science 79 (2016) 16 – 23, proceedings of International Conference on Communication, Computing and Virtualization (ICCCV) 2016. doi:https://doi.org/10.1016/j.procs.2016.03.004.
URL http://www.sciencedirect.com/science/article/pii/S1877050916001356

[29] L. Doerr, D. Fiala, M. Heigl, M. Schramm, Assessment simulation model for uncoupled message authentication, in: 2017 International Conference on Applied Electronics (AE), 2017, pp. 45–48. doi:10.23919/AE.2017.8053580.

[30] N. Ren, Q. S. Wang, C. Q. Zhu, Selective authentication algorithm based on semi-fragile watermarking for vector geographical data, in: 2014 22nd International Conference on Geoinformatics, 2014, pp. 1–6. doi:10.1109/GEOINFORMATICS.2014.6950830.

[31] J. Liu, F. R. Yu, C. H. Lung, H. Tang, Optimal combined intrusion detection and biometric-based continuous authentication in high security mobile ad hoc networks, IEEE Transactions on Wireless Communications (2009) 806–815doi:10.1109/TWC.2009.071036.

[32] A. Boudguiga, W. Klaudel, A. Boulanger, P. Chiron, A simple intrusion detection method for controller area network, in: 2016 IEEE International Conference on Communications (ICC), 2016, pp. 1–7. doi:10.1109/ICC.2016.7511098.

[33] M. Heigl, M. Schramm, L. Doerr, A. Grzemba, Embedded plug-in devices to secure industrial network communications, in: 2016 International Conference on Applied Electronics (AE), 2016, pp. 85–88. doi:10.1109/AE.2016.7577247.

[34] A. Bremler-Barr, H. Levy, Spoofing prevention method, in: Proceedings IEEE 24th Annual Joint Conference of the IEEE Computer and Communications Societies., Vol. 1, 2005, pp. 536–547 vol. 1. doi:10.1109/INFCOM.2005.1497921.

[35] Akamai, Akamais state of the internet: Q1 2014 report, Tech. rep., Akamai (2014).
URL http://www.akamai.com/dl/akamai/akamai-soti-q114.pdf

[36] M. Heigl, M. Aman, A. Fuchs, A. Grzemba, Securing industrial legacy system communication through interconnected embedded plug-in devices, in: Applied Research Conference (ARC), 2016, pp. 501–508.

[37] S. Salah, G. Maci-Fernndez, J. E. Daz-Verdejo, A model-based survey of alert correlation techniques, Computer Networks 57 (5) (2013) 1289 – 1317. doi:https://doi.org/10.1016/j.comnet.2012.10.022.
URL http://www.sciencedirect.com/science/article/pii/S1389128612004124

[38] E. A. Akkoyunlu, K. Ekanadham, R. V. Huber, Some constraints and tradeoffs in the design of network communications, in: Proceedings of the Fifth ACM Symposium on Operating Systems Principles, SOSP '75, ACM, New York, NY, USA, 1975, pp. 67–74. doi:10.1145/800213.806523.
URL http://doi.acm.org/10.1145/800213.806523

[39] R. Lupu, R. Badea, I. C. Mihai, Agent-based idmef alerting infrastructure for distributed intrusion detection and prevention systems: Design and validation, in: 2016 International Conference on Communications (COMM), 2016, pp. 281–284. doi:10.1109/ICComm.2016.7528341.

[40] E. Vasilomanolakis, S. Karuppayah, M. Mühlhäuser, M. Fischer, Taxonomy and survey of collaborative intrusion detection, ACM Computing Surveys 47 (2015) 1–33.

[41] G. Kumar, Evaluation metrics for intrusion detection systems - a study, International Journal of Computer Science and Mobile Applications (IJCSMA) 2 (11) (2014) 11 – 17.
URL `http://ijcsma.com/publications/november2014/V2I1105.pdf`

[42] P. García-Teodoro, J. Díaz-Verdejo, G. Maciá-Fernández, E. Vázquez, Anomaly-based network intrusion detection: Techniques, systems and challenges, Computers and Security 28 (1-2) (2009) 18–28. `doi:10.1016/j.cose.2008.08.003`.
URL `http://dx.doi.org/10.1016/j.cose.2008.08.003`

[43] M. Rezvani, Assessment methodology for anomaly-based intrusion detection in cloud computing, in: Journal of AI and Data Mining, Vol. 6, 2018, pp. 387–397. `doi:10.22044/jadm.2017.5581.1668`.
URL `http://jad.shahroodut.ac.ir/article_1087.html`

[44] V. Zieglmeier, S. Kacianka, T. Hutzelmann, A. Pretschner, A real-time remote IDS testbed for connected vehicles, in: CoRR, Vol. abs/1811.10945, 2018. `arXiv:1811.10945`.
URL `http://arxiv.org/abs/1811.10945`

[45] T. Pevný, Loda: Lightweight on-line detector of anomalies, in: Machine Learning, Vol. 102, 2016, pp. 275–304. `doi:10.1007/s10994-015-5521-0`.
URL `https://doi.org/10.1007/s10994-015-5521-0`

[46] N. Mouha, B. Mennink, A. van Herrewege, D. Watanabe, B. Preneel, I. Verbauwhede, Chaskey: An efficient mac algorithm for 32-bit microcontrollers, in: Selected Areas in Cryptography - SAC 2014, Springer International Publishing, Cham, 2014, pp. 306–323.

[47] A. R. Chowdhury, S. DasBit, Lmac: A lightweight message authentication code for wireless sensor network, in: 2015 IEEE Global Communications Conference (GLOBECOM), 2015, pp. 1–6. doi:10.1109/GLOCOM.2015.7417118.