# Flexible Multicast Authentication for Time-Triggered Embedded Control Network Applications

Christopher Szilagyi
ECE Department
Carnegie Mellon University
szilagyi@cmu.edu

Philip Koopman
ECE Department
Carnegie Mellon University
koopman@cmu.edu

## Abstract

*Security for wired embedded networks is becoming a greater concern as connectivity to the outside world increases. Protocols used in these networks omit support for authenticating messages to prevent masquerade and replay attacks. The unique constraints of embedded control systems make incorporating existing multicast authentication schemes impractical. Our approach provides multicast authentication for time-triggered applications by validating truncated message authentication codes (MACs) across multiple packets. We extend this approach to tolerate occasional invalid MACs, analyze our approach through simulated attacks, and give an upper bound on the probability of successful attack. This approach allows a tradeoff among per-packet authentication cost, application level latency, tolerance to invalid MACs, and probability of induced failure, while satisfying typical embedded system constraints.*

## 1. Introduction

Distributed embedded systems employing wired networks have numerous potential vulnerabilities. Anyone with physical access to the system, including the owner, can perform an attack to manipulate message traffic on the internal network. Increasing connectivity to external networks, such as the Internet or wireless networks, can also make wired embedded networks susceptible to attacks originating from those external networks, making security a more significant consideration in embedded control system design [14].

If an attacker can gain control over one of the nodes connected to a system, either through physical tampering or remotely subverting a gateway node, they gain access to the potentially safety-critical internal bus traffic of these systems. An attacker can eavesdrop on traffic, inject and modify messages, and can even perform DoS attacks [29]. By accessing this internal traffic, the attacker might, for example, engage the emergency brake in a car while it is traveling on a highway, unlock doors and start the engine, or shut off headlights while traveling at night.

Embedded control networks commonly use protocols such as Controller Area Network (CAN) [3], FlexRay [1], and Time-Triggered Protocol (TTP) [16]. Applications include distributed automotive, aviation, robotics, and industrial control systems. Safety, reliability, and cost have traditionally been the primary concerns in these systems, with security a minor concern. Most embedded control networks do not have any built in security to support authenticating nodes, encrypting data, restricting message types a node can send, or preventing Denial of Service (DoS) attacks.

In this paper, we expand upon our approach for authentication in time-triggered applications [28] which prevents both masquerade and replay attacks. Masquerade attacks [27] occur when a node sends a message in which it claims to be a node other than itself. This attack can be performed by broadcasting during another node's Time Division Multiple Access (TDMA) slot or by changing a message identifier value. Replay attacks [27] occur when a previously sent message is recorded and retransmitted by an attacker. Authentication allows a receiver to confirm the identity of a sender, typically via cryptographic mechanisms such as a Message Authentication Code (MAC) or a Digital Signature [27]. While wired embedded network protocols use error detection codes to verify message integrity, these codes can readily be forged, and are no substitute for strong cryptographic mechanisms.

Our previous work [28] provides multicast authentication in a single-hop wired embedded network, using truncated MACs. Truncating MACs allows the sender to place one small authenticator field per receiver in each message to allow authentication on a per-packet basis and provide tolerance to lost messages. It takes advantage of the time-triggered nature of many embedded control systems to authenticate a series of packets with consistent message values to gain confidence in both state-changing and reactive control messages. In this paper, we extend our approach to tolerate occasional invalid MACs (malicious and non-malicious) which could disrupt authentication of state-

Preprint from 2009 International Conference on Dependable Systems and Networks (DSN09)

changing messages. We also investigate cases where forging nonconsecutive reactive control messages within a period of time might lead to a successful attack. Finally, we provide analysis and experimental verification of the probability of maliciously induced failures for our approach.

In this paper Section 2 describes common wired embedded network constraints. Section 3 discusses multicast authentication using one MAC per receiver. Section 4 reviews related work. Section 5 describes our approach for authentication and Section 6 contains our analysis of the approach. Finally, Section 7 presents our conclusions.

## 2. Embedded network constraints

Distributed embedded networks are composed of a number of hardware Electronic Control Units (ECUs). These ECUs communicate via a network using a protocol such as CAN, FlexRay, or TTP to accomplish time-triggered communications. These protocols are among the most capable of those currently in use in wired embedded system networks. Many other protocols are even more resource constrained, but have generally similar requirements. We assume that embedded networks exhibit the following characteristics:

**Time-triggered** - In this paper, we consider only time-triggered applications. [15] defines a real-time system as time-triggered if all communications and processing activities are initiated at predetermined points in time from an a priori designated clock tick. Each node periodically broadcasts current values of state variables and sensor inputs to the rest of the network at designated times.

**Multicast communications** - Most distributed embedded networks are inherently multicast. All nodes connected to the network receive every packet. (In CAN, hardware performs message filtering at the receiver based on content.) Each packet includes the sender's identity, often implicitly through a message identifier (CAN; FlexRay) or time slot (TTP), but usually no explicit destination information. The configuration of the network is usually fixed at design time, with little or no run-time reconfiguration. Usually only a few nodes are attached to any network (commonly 32 or fewer).

**Resource limited nodes** - Processing and storage capabilities of nodes are often limited due to cost considerations. For example, the S12XD series, produced by Freescale [2], is a family of 16-bit microcontrollers designed for use in general automotive body applications. These microcontrollers provide up to 32 kilobytes of RAM, 512 kilobytes of flash memory, and four kilobytes of EEPROM, with a core operating frequency of 80 MHz. Flash memory is generally not written except for software updates, so EEPROM holds non-volatile application data. Buffering and storage for authentication consume space in RAM, which is far more expensive and scarce than flash memory in such systems. Authentication mechanisms which require large amounts of processing power or storage in RAM may not be feasible. More powerful ECUs are impractical for most nodes in the system, and many nodes are 8-bit ECUs with significantly smaller memories due to cost and power considerations.

**Small packet sizes** - Packet sizes are small in embedded network protocols when compared to those in enterprise networks. Packets have maximum data payload sizes as small as eight bytes in the case of CAN, with the largest payloads for FlexRay and TTP being 254 bytes and 236 bytes respectively. Cost, signal integrity, and network node synchronization concerns limit data rates to 1 Mbit/sec for CAN and 10 Mbit/sec for TTP and FlexRay. Low-cost embedded networks can be orders of magnitude slower than that. Authentication should incur minimal bandwidth overhead.

**Tolerance to packet loss** - Distributed embedded systems are subject to message blackouts from environmental disturbances such as interference from large electric motors. High quality cable shielding is often impractical due to cost, size, and weight limits. As such, authentication schemes must tolerate packet losses as part of normal system operation.

**Real-time deadlines** - In real-time systems, processes must complete within specified deadlines. Authentication of nodes must occur within a known time bound, with that bound being fast enough to match the physical time constants of the system being controlled (as fast as tens of milliseconds).

## 3. Multicast authentication with respect to embedded constraints

The multicast nature of distributed embedded communications makes authentication particularly challenging. Point-to-point cryptographic mechanisms, such as appending a MAC to a message using a single key shared across all nodes, do not provide adequate authentication. If more than two nodes share a key, any node which holds that key can masquerade as a different sender. For this reason, multicast authentication requires some form of key asymmetry, so that no node or group of colluding nodes can masquerade as another node. Sections 3.1 and 3.2 discuss the use of a single MAC per receiver and the limitations of this technique. Our approach, described in Section 5, validates truncated MACs over multiple messages.

### 3.1. One MAC per receiver

While typically avoided in enterprise networks where hundreds of receivers can require kilobytes of authentication data per message, using one MAC per receiver

can seem attractive for a wired embedded network having only tens of receivers. Sending one MAC per receiver is a simple extension of using shared secret keys, where each sender establishes a unique shared pair-wise key with every other node to provide asymmetric key possession. These MACs can be computed in milliseconds (or microseconds in hardware) for a small number of receivers. For each transmitted message, the sender appends a MAC for each distinct receiver. A receiver would know that a message with a valid MAC could only have come from the sender, because those two nodes uniquely share a secret key (and the receiver knows it did not send the message).

However, even for a small number of receivers, the bandwidth overhead for full-size MACs makes this approach infeasible for most embedded networks. The total bit length of the MACs can be tens to hundreds of times greater than the size of a single message. (Consider a common situation in which a message reports whether a switch is "on" or "off," requiring a one-bit data payload with perhaps thousands of bits of authenticator information.) Data could only be sent rarely in this scheme, because most bandwidth would be spent on authentication. Moreover, message size constraints would require fragmenting MACs across multiple messages, and packets in many protocols are not retransmitted immediately if corrupted. If a packet containing even a small part of a receiver's authenticator is lost, the receiver would have to wait up to an entire set of message rounds until a new value and authenticator is broadcast. This delay may not be tolerable in a real-time system.

### 3.2. Possible improvements

One possible improvement is to compute these MACs over a set of several messages to amortize the bandwidth cost over that set. The sending node broadcasts each message in the set in its respective time slot. Once the set of messages has been transmitted, the sender computes the MACs over the set and in following time slots broadcasts one MAC per receiver. While this amortizes the overhead over many packets, it induces even longer latency to authentication in the event that a message value or authenticator is lost.

Another possibility is to compute the set of MACs once every $n^{th}$ time a message type is broadcast. This improves loss tolerance since fewer packets contain authenticators, and the bandwidth cost for authentication can be made arbitrarily small. Scheduling can be simplified by taking turns sending authenticators for different receivers (for example, sending only one receiver's authenticator in turn with each message). But, any single receiver must wait for up to $n$ messages to arrive to see one with an authenticator it knows how to check. The receiver cannot determine whether the $n-1$ other messages are an attacker's forgeries or not. Additionally, if the $n^{th}$ packet or its authenticator is lost in transmission, the receiver suffers a delay of another full set of $n$ messages and the authenticators.

Schemes using one MAC per receiver are simple and computationally fast, but the poor ratio of message data to authentication data, and unacceptable delays due to losses must be addressed. In this paper we discuss and improve upon an approach which solves these issues by exploiting time-triggered communications.

## 4. Related Work

### 4.1. Existing multicast authentication schemes

Public key cryptography using digital signatures is another asymmetric approach. While this could provide strong source authentication, the processing overhead makes it impractical for a resource constrained node to compute digital signatures for real time control. Pagers and Palm Pilots can take several seconds to compute a 512 bit RSA signature in resource constrained nodes [4]. Several schemes suggest amortizing the cost of the digital signature over several packets [18][21][25][30]. But, a node would have to amortize the cost over several hundred messages for this to be effective.

Schemes using one-time digital signatures [8][10][22] allow senders to sign messages much faster than with traditional digital signatures by using one-way hash functions, at the expense of increased message sizes. Unfortunately, one-time digital signatures can incur several kilobytes of authentication data per message. This makes them impractical for embedded networks with small packet sizes and time-triggered communication, even if amortized over many packets.

Canetti et al. [5] suggest a scheme which appends $k$ one-bit MACs to each message, computed using $k$ different keys. The keys are distributed amongst receivers such that at least $w$ receivers must conspire to forge a message. While this is more efficient than using one MAC per receiver, it is vulnerable to collusion by multiple nodes that together can masquerade as some other node. Mitigating collusion can require hundreds or thousands of authentication bits per message.

TESLA [23] uses time-delayed release of keys to provide asymmetry. By releasing keys at a pre-specified interval after a MAC is released, receivers can confirm the authenticity of the data from a sender. The released keys are computed using one-way hash chains, but require significant memory space. μTESLA [24], a version of TESLA for resource constrained sensor networks, limits the number of authenticated senders and utilizes a base station for communications. A base station is often cost-prohibitive for distributed embedded real-time control systems, which use peer-to-peer wired networks. An existing node, such as an embedded gateway, might act as a base station, but

would be an undesirable single point of failure for the entire network. A fully distributed approach is best.

### 4.2. Embedded network authentication

This work builds upon [28], which provides multicast authentication on a per-packet basis in time-triggered applications, using one truncated MAC per receiver for wired networks.

Other approaches such as SPINS [24] and TinySec [13] apply security to resource constrained wireless sensor networks. However, those approaches are specifically designed for use in wireless networks, which have significantly different constraints. Secure aggregation [12][26] focuses on aggregation of data from multiple sensors in close geographic proximity rather than time-triggered messages in temporal proximity.

Morris and Koopman [19] identify the potential for masquerade failures to cause accidental or malicious failures, via non-critical nodes masquerading as higher criticality nodes. They propose the use of countermeasures of varying strengths to prevent masquerading failures between nodes of varying criticality. Their approach assumes non-malicious software faults or attacks from a cryptologically unsophisticated attacker. Fault tolerance mechanisms are not necessarily secure against malicious masquerade or replay attacks. Masquerade prevention for safety-based systems typically uses bus guardians or a symmetric key shared among all trusted nodes. Compromise of a single node would permit an attacker to masquerade as any system node.

Wolf et al. [29] provide an overview of the security vulnerabilities of various in-vehicle network protocols including Local Interconnect Network (LIN), Media Oriented System Transport (MOST), CAN, and FlexRay. These vulnerabilities primarily focus upon DoS attacks intended to disable networks. Additionally, they state the need for confidentiality and authentication. Wolf et al. suggest the use of digital signatures or the asymmetric MAC scheme proposed in [5] for authenticating sent packets along with gateways between individual in-vehicle networks. These authentication schemes may not be suitable for some distributed embedded networks, as discussed in Section 4.1.

There have been several publications demonstrating attacks on the authenticity of messages and nodes in embedded networks. Nilsson and Larson [20] detail the actions which an attacker might take, and demonstrate masquerade attacks on CAN using simulation. Hoppe et al. [11] and Lang et al. [17] demonstrate a combination of eavesdropping and replay attacks on CAN.

Lastly, Chávez et al. [6] propose using RC4 encryption to provide confidentiality on CAN buses. They dismiss authentication and non-repudiation as unnecessary in these networks, under the assumption that message identifiers and error detection provide sufficient confirmation of the sender's identity. Our work relaxes this assumption by assuming that sender identity can be forged, for example as discussed in [20].

## 5. Criticality-based authentication

Our approach provides multicast authentication on a per-message basis in time-triggered applications, using one truncated MAC of just a few bits per receiver. Authentication of both state-changing messages and reactive control messages is accomplished by validating these truncated MACs across multiple packets. In time-triggered applications, each node periodically broadcasts the current state of each of its state variables and sensor inputs to the rest of the network. Information is often broadcast faster than the rate at which receivers must act upon the data in their control loops, allowing authentication of messages over a series of packets containing consistent values. This faster rate also gives the system a degree of resilience to unexpected operating situations and packet losses even with no authentication. This resilience to packet losses carries over to our authentication approach, because all information required to validate a single packet is self-contained.

### 5.1. Message types

We distinguish between two types of messages: state-changing and reactive control. State-changing messages cause transitions within finite state machines in the system design, or cause discrete, discontinuous output changes in actuators. In our approach, to authenticate state-changing messages nodes must receive a predefined number of correctly authenticated packets with consistent message values directing the state change before executing the action. A set of state-changing packets values are *consistent* if all data values are equal, or all are within a predefined range in which each would trigger the same state-change.

Our approach does not require the sender to transmit extra messages. Instead, our approach takes advantage of periodic transmissions of current state values, enabling a tradeoff between application level latency, per-packet authentication cost, and probability of induced failures. The number of required packets and authentication bits per packet depends upon the criticality of the state change, as discussed later.

Reactive control system messages cause updates to continuous or ordered values in network nodes running feedback control loops. These loops often contain a low pass filter to actuator changes (implicit or explicit), such as physical inertia. Again, a receiver performs authentication over packets with consistent values. Reactive control message values are *consistent* if they pass standard validity or sanity checks (such as input bounds checking), which is a less stringent criterion than that for state-changing messages. Authenticating

each consistent message value allows the receiver to perform per-message validation of messages. The system will tolerate some small number of forged messages because of physical damping, requiring the attacker to forge multiple messages before an unsafe output can occur. So long as each reactive control message contains enough authenticator bits, the probability of successfully forging such a large set of messages can be made sufficiently low. This enables a tradeoff between per-packet authentication costs and probability of induced failure.

## 5.2. Assumptions

This scheme relies upon several assumptions:

- Packets are transmitted at a rate fast enough for a receiver to authenticate multiple consistent values for a message type within system deadlines.
- Each sender has sufficient computational resources to compute one MAC per receiver per packet that is sent. The required computational resources depend on the MAC function used.
- The number of bits in a packet's data payload is greater than the number of receivers of a packet. This allows authenticators for each receiver in the packet, leaving room for the message value.
- Nodes use existing cryptographic one-way hash functions, such as SHA-256, and MAC functions to implement authentication ([27] includes examples). We assume the underlying cryptographic primitives are secure. We do not rely on specific MAC or one-way hash functions to implement our scheme.
- A certification authority exists to assign key material to components when they are manufactured.
- The network configuration is fixed; nodes are not installed or uninstalled on the fly.
- Nodes remain synchronized to the nearest message round.

## 5.3. Attacker model

We assume an attacker can gain access to the system through a gateway connection to an external network, malicious insider code, physical access to network lines, or tampering with nodes. They may own the device being attacked. We consider an active attacker model [27] in which an attacker may modify, inject, drop, or eavesdrop upon network traffic.

Attackers accessing the network through corrupted nodes will have access to the key material in those nodes. An attacker must not be able to masquerade as any node they do not already control to perform a successful attack, except by random chance.

We will assume an attacker is aware of existing error detection mechanisms along with the network schedule, and is capable of injecting well-formed packets in valid time slots. This constrains an attacker to one forgery attempt per valid time slot in a TDMA network such as TTP or FlexRay, since transmitters are only permitted to transmit a single packet per time slot in a time-triggered application. For the purpose of analysis, we consider the worst case scenario in which a successful attack depends solely on fooling a single receiver. In practice, isolating receivers may be difficult if strong existing fault containment mechanisms such as group membership are used.

Additionally, we consider the effects of packets containing invalid MACs (malicious and non-malicious) which might disrupt authentication. We do not consider full DoS attacks intended to prevent delivery of all network traffic, because as discussed by Wolfe et al. [29], there are numerous existing vulnerabilities in these networks to that type of attack, and our scheme does not attempt to address these.

## 5.4. Authentication process

**5.4.1. Key initialization and replay protection.** Each node is programmed with a public and private Diffie-Hellman [7] key pair (digitally signed by a trusted certification authority, such as the manufacturer) and the certification authority's public key. Upon installation or replacement, each node uses these keys to establish a shared secret key with each other node. Because of the pair-wise shared keys, an attacker cannot masquerade as any node other than the one compromised. All nodes wired to the network are known at design time, and key establishment costs are incurred once at installation.

Replay protection is provided using a protocol for securely synchronizing time or TDMA round number between nodes such as the Secure Pair-wise Synchronization protocol [9], once pair-wise keys are established. This can provide synchronization on the order of microseconds to ensure freshness of messages for each message round, which can be tens to hundreds of milliseconds. Global synchronization is not needed, since only pairs of nodes share each secret key. We include current time or TDMA round number along with the secret key as inputs to a cryptographically secure MAC function. Synchronized time values must not roll over for some acceptably long period of time. This prevents the attacker from predicting the MACs over this period of time even for identical data values.

**5.4.2. Run-time message generation.** When a node sends a packet, it computes a MAC for each distinct receiving node in the network over the message value, packet header, and the current time using the pair-wise shared secret key. Each MAC is truncated down to just a few bits, and appended to the message value. By only using a few bits, one MAC per receiver can be placed in a single packet, as illustrated in Figure 1. The size of each truncated MAC could be as little as one bit per

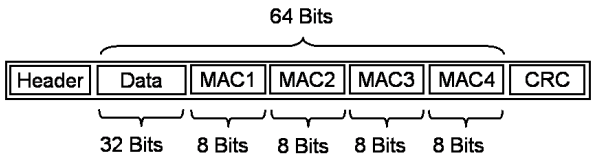


**Figure 1. Example packet containing 32 bits of data and four 8-bit MACs, for four receivers.**

MAC. All authentication for each packet is fully contained within the data payload of that packet, allowing each packet to be verified independently.

**5.4.3. State-changing message verification.** A receiver authenticates state-changing messages by authenticating a set of packets. The node keeps an authentication *history buffer* for each message type used. Each packet is authenticated individually when received, and the receiver stores the results ("valid" or "invalid") for the $n$ most recent packets in the history buffer. A receiver considers a packet to be valid if it contains correct authentication and error detection fields. It is invalid if the error detection field is correct and the authentication field is incorrect. Any packets containing an incorrect error detection field are invalid, and are omitted from the history buffer.

The state change occurs when at least $k$ out of the past $n$ time-triggered packets have consistent values and are valid. Assume that each packet contains $b$ authentication bits per receiver. State changes occur as soon as the $k^{th}$ packet with a consistent message value has been validated. While it is likely that an attacker will be able to forge a single packet since we use just a few authentication bits per MAC, it is unlikely that they will be able to forge so many within the history of the buffer as to cause a successful masquerade attack. An attacker can successfully forge at least $k$ of a set $n$ packets with a binomial probability of:

$$P_A = \sum_{i=k}^{n} \binom{n}{i} (2^{-b})^i (1-2^{-b})^{n-i} \quad (1)$$

Allowing state changes to occur after validating a subset of MACs in the history buffer grants this approach a degree of tolerance to interspersed invalid MACs. Without this tolerance, an attacker can increase message latency or prevent authentication altogether while remaining undetected by occasionally injecting invalid packets. Packets with a correct CRC but invalid MAC might also be caused by non-malicious faults. For example, if the sender's and receiver's notions of time differ due to a temporary internal fault, the receiver would see an invalid MAC. Additionally, some message corruptions might be missed by error detection mechanisms, so occasional invalid MACs might result from transmission errors.

In applications which do not require tolerance to invalid MACs or require a very low probability of successful attack, the receiver waits for a set of consecutively validated MACs. In this case, $k$ is equal to $n$. The probability of forging $n$ consecutive packets is:

$$P_A = 2^{-nb} \quad (2)$$

This approach for authenticating state-changing messages enables the system designer to perform a tradeoff among per-packet authentication cost, application level latency, tolerance to invalid MACs and probability of an induced failure. Based upon the criticality of the message, the designer trades increased bandwidth and latency for lower probability of failure, and trades increased tolerance to invalid MACs for increased risk of induced failure. Additionally, there is a limit on the number of required packets, based upon the maximum tolerated latency for authentication, how many packets are expected to contain consistent message values (depending upon network speed), and the bandwidth available for authentication bits in packets.

**5.4.4. Reactive control message verification.** Unlike state-changing message verification, nodes running feedback control loops authenticate each message packet as it arrives. Each authenticated message causes an immediate change in actuator outputs. Packets containing detected transmission errors or incorrect MACs are discarded without updating outputs. An actuator might cause an unsafe situation if it accepts too many successfully forged message values commanding it to an unsafe position within a time period, even if it receives valid message values within that period. We consider the case where at least $k$ messages must be successfully forged out of the $n$ most recently received messages to force the system to an unsafe state. Values for $n$ and $k$ depend on the characteristics of the system. More complex control systems in which messages cause varying amounts of actuation depending upon their value and exact timing require further analysis and are beyond the scope of this paper.

For reactive control messages, the receiver does not explicitly retain an authentication history buffer in memory, but relies instead upon a damped response to messages. In order to successfully attack the system, an attacker must individually forge at least $k$ out of the $n$ most recent messages sent to the receiver. This gives the probability of a successful undetected attack in equation (1). For actuators which require a set of $n$ consecutive messages to reach an unsafe output, equation (2) describes the probability of a successful attack.

This approach also supports trading increased per packet authentication cost for reduced probability of induced system failure. The designer selects the number of bits per packet $b$ based upon the amount of physical change produced per message, such that the

probability of system failure is considered acceptable.

**5.4.5. Tolerance to packet loss.** If a single packet is lost or corrupted due to an error during transmission, the receiver simply ignores that packet. Ignored packets do not disrupt authentication because receivers authenticate each packet individually based on data is fully contained within that packet.

## 6. Analysis

In this section we discuss characteristics of our approach and experimental results of simulated attacks. Per our attack model, an attacker may insert or modify packets in valid time slots for a particular message type. Computing the MAC over the pair-wise synchronized time or TDMA round number ensures freshness of messages. At best, an attacker may only insert a packet with a randomly generated MAC once per valid time slot. To be conservative in our analysis, the attacker performs masquerade attempts against a single isolated receiver, so an attacker only needs to guess one truncated MAC per packet.

We have experimentally confirmed the probability of successful forgery attacks against our approach using a software simulation written in C. In our simulation, an attacker node continually sends packets containing a known message value and randomly generated MAC values to the receiver. The receiver node verifies the packet using HMAC-SHA-256 and retains a history buffer of the *n* most recent authentication results. Once the receiver counts a sufficient number of valid MACs in its history buffer, the simulator records an *attack event* and the number of attempted forgeries before the successful attack occurred.

We simulated attacks on state-changing and reactive control messages for both authentication of consecutive packets and authentication of a fraction of packets in a history buffer. Attacks on state-changing messages were considered to be successful once the attacker forced a state change, and further packet forgeries were applied to the next state change after clearing the history buffer. For reactive control messages, successful attack events were recorded as long as the most recent packets contained a sufficient number of valid MACs, regardless of authentication history.

We measured the number of successful attack events over a period of time long enough to record at least one hundred successful attack events per data point. We computed the *successful attack rate* as average successful attack events per message round and compared this rate to the probability of successful attack defined in equations (1) and (2) in Section 5.4.3. From our results we confirmed that equations (1) and (2) can be used as upper bounds on the probability of successful attacks on our approach. These equations predict the required number of packets and authentication bits per packet to achieve a desired failure rate and tolerance to invalid MACs for the system.

### 6.1. Authenticating consecutive packets

Figure 2 shows the simulated successful attack rate on both state-changing and reactive control message types, using a fixed history buffer size of four packets containing one to six authentication bits per packet. As more bandwidth is devoted to authentication, the successful attack rate decreases exponentially.

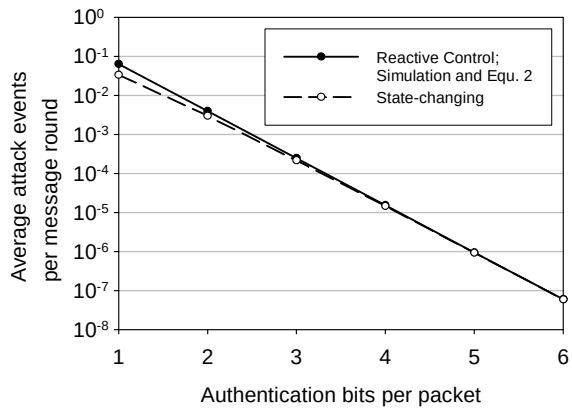


**Figure 2. Simulated successful attack rates for four consecutive messages.**

The successful attack rates in Figure 2 should be no greater than the probability of successful attack defined by equation (2). As expected, the successful attack rate for reactive control messages matches equation (2) since simulated attacks were counted regardless of previous authentication history. (Equation (2) is indistinguishable from the simulated reactive control successful attack rate if plotted on Figure 2.)

The successful attack rate for state-changing messages is less than the rate for reactive control messages because successful attacks are likely to come in bursts of consecutive reactive control messages containing few authentication bits. A forgery attempt on the packet after an initial attack event has a better probability of prolonging the attack in comparison to forging a full set of *n* packets to initiate a successful attack. The simulated successful attack rate for state-changing messages is approximately a factor of $(1\text{-}2^{-b})$ less than the rate for reactive control messages, because we assume the history buffer is flushed after a state change.

With more bits per packet, the likelihood of successful attacks occurring on successive reactive control messages decreases, as indicated by the converging rates in Figure 2. Thus, we can use equation (2) as a conservative upper bound on the successful attack rate for both reactive control and state-changing messages.

Typical requirements for acceptable failure rates in systems containing wired embedded networks might be defined at $10^{-3}$/hr, $10^{-6}$/hr, or $10^{-9}$/hr of undetected mes-

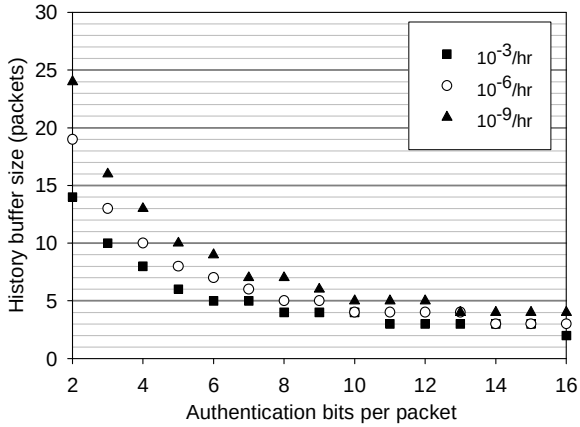


**Figure 3. Minimum authentication bits per packet and history buffer size required to authenticate to failure rates at 1000 packets per second.**

sage errors depending on the severity of the failure. An induced failure from a masquerade attack should occur no more often than the required rate of failure. Figure 3 shows the minimum number of messages in the history buffer for a given number of authentication bits per message to achieve an expected successful attack rate of $10^{-3}$/hr, $10^{-6}$/hr, or $10^{-9}$/hr. The number of packets and bits were obtained using the three successful attack rates as expected values for one forgery attempt per millisecond over the course of an hour, each succeeding with probability given by equation (2).

## 6.2. Authenticating nonconsecutive packets

If we permit interspersed invalid MACs in the authentication history buffer, we gain tolerance to some non-malicious faults and malicious attempts to disrupt authentication of state-changing messages. But increasing this tolerance also increases the probability of an induced failure. Attacks may succeed against some control systems if the attacker forges some fraction of the most recent reactive control messages. As this fraction decreases, the probability of induced failure increases.

Figure 4 shows the simulated successful attack rate on state-changing and reactive control message types requiring two successful forgeries out of four packets, each containing one through six authentication bits. As the number of bits per packet for authentication increases, the probability of a successful attack decreases exponentially.

The successful attack rate on reactive control messages in Figure 4 matches equation (1) because attack events were counted regardless of previous authentication history. (Equation (1) is indistinguishable from the Figures 4 and 5.) The successful attack rate for reactive control messages is greater than that for state-changing messages because successful attacks on reactive control messages can persist as long as the most recent *n*

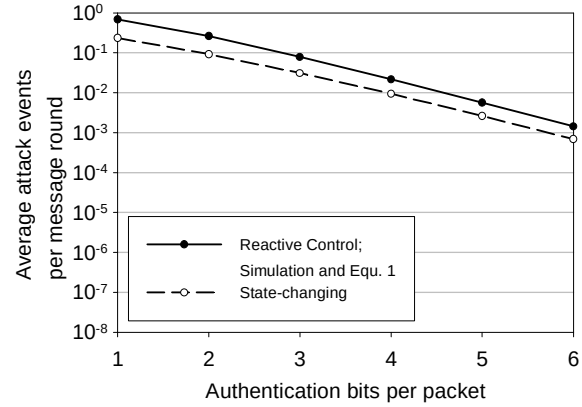


**Figure 4. Simulated successful attack rate for two out of four messages.**

packets contain *k* valid MACs. The difference between lines in Figure 4 is greater than the difference between lines in Figure 2 because there are multiple combinations of successful forgeries in the most recent packets which can cause successful attacks to persist. We do not attempt to provide an equation due to the complexity of the combinations. Rather, we use equation (1) as conservative upper bound for both message types.

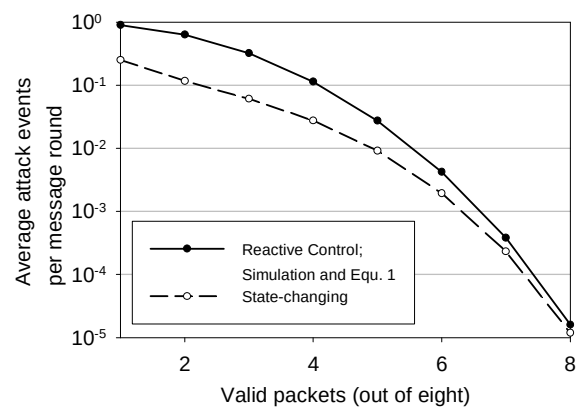


**Figure 5. Simulated successful attack rates for a history buffer of eight packets with two authentication bits each.**

Figure 5 illustrates how the difference between simulated successful attack rates for reactive control and state-changing messages changes as the number of required successful forgeries is varied for a buffer of eight packets each containing two authentication bits. With a lower fraction of required valid packets, there are more possible combinations which can cause a successful attack to persist for reactive control message types, causing a greater successful attack rate.

Figure 6 illustrates tradeoffs between history buffer size and authentication bits per packet needed for expected successful attack rates of $10^{-3}$/hr, $10^{-6}$/hr, or $10^{-9}$/hr, requiring all but two valid MACs. The number of packets and bits were obtained using the three suc-

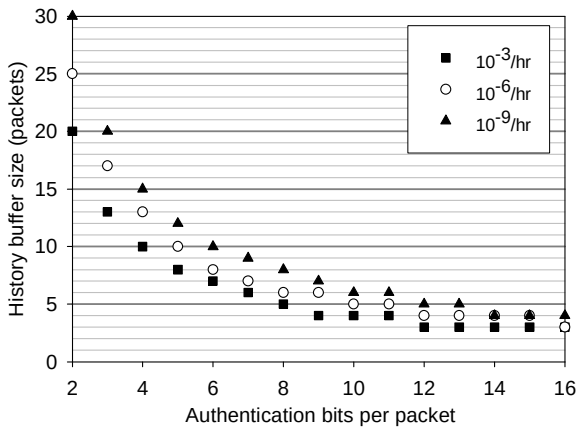


**Figure 6. Minimum authentication bits per message and history buffer size required to authenticate to failure rates at 1000 messages per second given two invalid packets in the buffer.**

cessful attack rates as expected values for one forgery attempt per millisecond over the course of an hour, each succeeding with probability of equation (1). For example, with four authentication bits per message, if all packets in a history buffer must be valid, the history buffer must include at least the last ten packets to authenticate to $10^{-6}$/hr (Figure 3). If all but two packets must be forged in the history buffer, then the history buffer must include the past thirteen packets (Figure 6).

### 6.3. Limitations

The rate of successful attacks via brute force guessing will be higher than an approach which authenticates an entire frame of packets all at once with one MAC (assuming number of total authentication bits is equal). Consider the case where $n$ consecutive packets each containing $b$ authentication bits are required to authenticate a state change. In our approach, the attacker has a probability of $2^{-nb}$ of successfully attack per attempt, where each attempt requires a single new packet. Next, consider the case where the same frame of $n$ packets (causing the exact same effect) is authenticated all at once using a single MAC containing $nb$ bits. The attacker has a probability of success of $2^{-nb}$ per attempt, however each attempt now requires sending all $n$ packets every time. Thus, on a per-packet basis, attack events are expected to occur $n$ times more frequently when using our approach compared to a single frame.

This limitation can be addressed in several ways. First, the system designer can add logic in a receiver to detect a large number of invalid packets as an intrusion attempt. During a brute force guessing attack, the receiver will get many invalid packets before a successful attack is likely. Second, this factor can be reduced by using a smaller buffer size. Finally, our approach can be slightly modified so the receiver waits for a full set of $n$ packets to arrive before committing to an action. The receiver authenticates all $n$ packets at once, then clears its history buffer. If the receiver gets too many invalid packets within one history buffer, it simply waits until $n$ packets have been received before clearing the history buffer and listening for a new set of packets. This takes advantage of the loss tolerance of our approach with a worst case latency increase of $n$ messages rounds. This technique extends to allowing invalid MACs by having the receiver reset its history buffer only after receiving all $n$ packets regardless of whether enough of them contained valid MACs.

Additionally, in our simplified model of reactive control systems an attack may continue with relatively high probability once it has successfully started as discussed in Section 6.1. To prevent this, each packet must incorporate enough bits to keep the probability of individual message forgery at an acceptable level. As this implicit history buffer becomes longer with fewer authentication bits per packet, the probability of prolonged attack increases. This can be mitigated by using more bits per packet.

Our approach has additional limitations. Each message requires the computation of one MAC per receiver. Hardware support for cryptographic computations is desirable and might be incorporated directly as part of hardware support for the communications protocol. This suggests a research opportunity for fast, inexpensive MAC functions producing small outputs.

A scalability limit is that the number of authentication bits per packet grows linearly with the number of receivers. This might be mitigated by omitting MACs for receivers that don't need to use the value of a particular message.

Our method assumes time-triggered applications. It relies on the periodic broadcasts of current values of state variables, and the limitation of one packet per TDMA slot. Other approaches are needed for event-triggered networks to provide strong authentication for each event. Our system provides advantage to the degree that messages are transmitting over-sampled data.

Lastly, our approach does not tolerate complete DoS attacks. Allowing intermittent invalid packets within a history buffer is a useful technique for tolerating stealthy attacks or non-malicious faults. But if an attacker floods a network with invalid packets, a receiver must assume that the network has suffered a permanent failure and take appropriate action.

## 7. Conclusions

In this paper we build upon an approach to authenticate time-triggered communications by validating truncated MACs across multiple packets. Our approach enables per-message authentication of reactive control messages and delayed authentication of state changes at a slight increase in the probability of induced failures

(when compared to using a single strong MAC). We tolerate occasional packets with invalid MACs interspersed with valid MACs, consider cases where forging nonconsecutive reactive control messages leads to successful attacks, and provide a conservative upper bound on the probability of successful attack. This approach enables a tradeoff among per-packet authentication cost, application level latency, tolerance to invalid MACs and probability of induced failures to provide flexibility for system designers.

## 8. Acknowledgements

This research was funded in part by General Motors through the GM-Carnegie Mellon Vehicular Information Technology Collaborative Research Lab.

## 9. References

[1] FlexRay Consortium. FlexRay Communications System Protocol Specification, Version 2.1, Rev. A, Dec. 2005.

[2] Freescale Semiconductor. S12XD Product Summary Page. Accessed Dec. 2008 at http://www.freescale.com/.

[3] R. Bosch GmbH, CAN Specification, Ver. 2, Sep. 1991.

[4] M. Brown, D. Cheung, D. Hankerson, J. L. Hernandez, M. Kirkup, and A. Menezes. PGP in constrained wireless devices. In *SSYM'00: Proc. of the 9th Conf. on USENIX Security Symposium*, pp. 19–34, 2000.

[5] R. Canetti, J. Garay, G. Itkis, D. Micciancio, M. Naor, and B. Pinkas. Multicast security: a taxonomy and some efficient constructions. In *INFOCOM '99: Proc. 18th Annual Joint Conf. of the IEEE Computer and Communications Societies*, vol. 2, pp. 708–716. IEEE, 1999.

[6] M. Chavez, C. Rosete, and F. Henriquez. Achieving Confidentiality Security Service for CAN. In *CONIELECOMP '05: Proc. of the 15th Int'l Conf. on Electronics, Communications and Computers*, pp. 166–170, 2005.

[7] W. Diffie and M. Hellman. New directions in cryptography. *IEEE Transactions on Information Theory*, vol. 22, pp. 644-654, 1976.

[8] S. Even, O. Goldreich, and S. Micali. On-line/off-line digital signatures. In *CRYPTO '89: Proc. on Advances in Cryptology*, pp. 263–275. Springer-Verlag, 1989.

[9] S. Ganeriwal, S. Čapkun, C.-C. Han, and M. B. Srivastava. Secure time synchronization service for sensor networks. In *WiSe '05: Proc. of the 4th ACM Workshop on Wireless Security*, pp. 97–106. ACM, 2005.

[10] R. Gennaro and P. Rohatgi. How to Sign Digital Streams. In *CRYPTO '97: Proc. of the 17th Annual Int'l Cryptology Conf. on Advances in Cryptology*, pp. 180–197. Springer-Verlag, 1997.

[11] T. Hoppe and J. Dittman. Sniffing/Replay Attacks on CAN Buses: A simulated attack on the electric window lift classified using an adapted CERT taxonomy. In *2nd Workshop on Embedded Systems Security (WESS)*, 2007.

[12] L. Hu and D. Evans. Secure Aggregation for Wireless Networks. In *Proc. of the 2003 Symp. on Applications and the Internet Workshops*, pp. 384–394. IEEE, 2003.

[13] C. Karlof, N. Sastry, and D. Wagner. TinySec: a link layer security architecture for wireless sensor networks. In *SenSys '04: Proc. of the 2nd Int'l Conf. on Embedded Networked Sensor Systems*, pp. 162–175. ACM, 2004.

[14] P. Koopman, J. Morris, and P. Narasimhan. Challenges in Deeply Networked System Survivability. *NATO Advanced Research Workshop on Security and Embedded Systems*, pp. 57–64, 2005.

[15] H. Kopetz. *Real-Time Systems: Design Principles for Distributed Embedded Applications*. Kluwer Academic Publishers, Norwell, MA, USA, 1997.

[16] H. Kopetz and G. Grunsteidl. TTP - A time-triggered protocol for fault-tolerant real-time systems. In *Proc. of the 23rd Int'l Symposium on Fault-Tolerant Computing*, pp. 524–533, 1993.

[17] A. Lang, J. Dittman, S. Kiltz, and T. Hoppe. Future Perspectives: The car and its IP address - A potential safety and security risk assessment. In *Proc. of the 26th Int'l Conf. on Computer Safety, Reliability and Security (SAFECOMP)*, pp. 40-53. Springer-Verlag, 2007.

[18] S. Miner and J. Staddon. Graph-Based Authentication of Digital Streams. In *SP '01: Proc. of the 2001 IEEE Symp. on Security and Privacy*, pp. 232–246.

[19] J. Morris and P. Koopman. Critical Message Integrity Over A Shared Network. *5th IFAC Int'l Conf. on Fieldbus Systems and their Applications*, pp. 145-151, 2003.

[20] D. Nilsson and U. Larson. Simulated Attacks on CAN Buses: Vehicle Virus. *5th IASTED Asian Conf. on Communication Systems and Networks*, 2008.

[21] J. M. Park, E. K. P. Chong, and H. J. Siegel. Efficient Multicast Packet Authentication Using Signature Amortization. In *SP '02: Proc. of the 2002 IEEE Symposium on Security and Privacy*, pp. 227–240. IEEE, 2002.

[22] A. Perrig. The BiBa one-time signature and broadcast authentication protocol. In *CCS '01: 8th ACM Conf. on Computer and Comm. Security*, pp. 28–37, 2001.

[23] A. Perrig, R. Canetti, J. Tygar, and D. Song. The TESLA Broadcast Authentication Protocol. *RSA CryptoBytes*, vol. 5, pp. 2-13, 2002.

[24] A. Perrig, R. Szewczyk, J. D. Tygar, V. Wen, and D. E. Culler. SPINS: security protocols for sensor networks. *Wireless Networks*, vol. 8(no. 5):pp. 521–534, 2002.

[25] A. Perrig, J. D. Tygar, D. Song, and R. Canetti. Efficient Authentication and Signing of Multicast Streams over Lossy Channels. In *SP '00: Proc. of the 2000 IEEE Symposium on Security and Privacy*, pp. 56–73, 2000.

[26] M. Raya, A. Aziz, & J. Hubaux. Efficient secure aggregation in VANETs. In *VANET '06: 3rd Int'l Workshop on Vehicular Ad Hoc Networks*, pp. *67–75. ACM, 2006*.

[27] Schneier. *Applied Cryptography (2nd ed.): Protocols, Algorithms, and Source Code in C*. John Wiley & Sons, Inc., New York, NY, USA, 1995.

[28] C. Szilagyi and P. Koopman. A flexible approach to embedded network multicast authentication. In *2nd Workshop on Embedded Systems Security (WESS)*, 2008.

[29] M. Wolf, A. Weimerskirch, and C. Paar. Security in Automotive Bus Systems. *Workshop on Embedded Security in Cars*, 2004.

[30] C. K. Wong and S. S. Lam. Digital Signatures for Flows and Multicasts. In *ICNP '98: Proc. of the 6th Int'l Conf. on Network Protocols*, pp. 198–209. IEEE, 1998.