Flexible, Extensible, and Efficient VANET Authentication

Ahren Studer[†], Fan Bai[§], Bhargav Bellur[§] & Adrian Perrig[†]
[†] Carnegie Mellon University [§] General Motors
{astuder, perrig}@cmu.edu {fan.bai, bhargav.bellur}@gm.com

ABSTRACT

The authentication of VANET messages continues to be an important research challenge. Although much research has been conducted in the area of message authentication in wireless networks, VANETs pose unique challenges, such as real-time constraints, processing limitations, memory constraints, requirements for interoperability with existing standards, extensibility and flexibility for future requirements, etc. No currently proposed technique addresses all of these requirements.

After analyzing the requirements for viable VANET authentication, we propose a modified version of TESLA, TESLA++, which provides the same computationally efficient broadcast authentication as TESLA with reduced memory requirements. To address the range of needs within VANETs we propose a new hybrid authentication mechanism, VANET Authentication using Signatures and TESLA++ (VAST), that combines the advantages of ECDSA signatures and TESLA++. ECDSA signatures provide fast authentication and non-repudiation, but are computationally expensive. TESLA++ prevents memory and computation-based Denial of Service attacks. We analyze the security of our mechanism and simulate VAST in realistic highway conditions under varying network and vehicular traffic scenarios. Simulation results show that VAST outperforms either signatures or TESLA on their own. Even under heavy loads VAST is able to authenticate 100% of the received messages within 107ms.

1. INTRODUCTION

Within the next decade, vehicles will be equipped with Dedicated Short Range Communication (DSRC) capabilities to provide a means for a Vehicular Ad Hoc Network (VANET) where vehicles' On-Board Units (OBUs) communicate wirelessly with other vehicles' OBUs or Road Side Units (RSUs) [2]. Vehicle manufacturers and federal entities intend to leverage these VANETs to make roadways safer and improve the driving experience through a number of safety, convenience, and commercial applications.

For VANET applications to operate safely, an authentication framework is necessary to help identify valid participants, ensure participants are who they claim to be, and prevent malicious parties from modifying messages. Without an authentication framework, attackers could physically or financially harm other drivers. For example, malicious parties could broadcast spurious data and cause vehicular accidents–accidents which otherwise would have been avoided if VANETs were not in use. Malicious parties could pose as electronic toll booths and steal drivers' financial information.

The current IEEE 1609.2 standard for secure VANET communication proposes the use of the Elliptic Curve Digital Signature Algorithm (ECDSA) for signatures to verify messages [7]. Prior work has shown that the verification of a single ECDSA signature requires 7ms of computation on proposed OBU hardware [15]. However, an attacker can send an invalid signature in a fraction of that time. This imbalance between time needed to process and time needed to receive gives rise to Denial of Service (DoS) attacks. An attacker could use a fraction of the DSRC bandwidth to flood receivers with invalid signatures which will take much longer to process. Without a more efficient authentication mechanism, attackers could cripple a VANET.

TESLA appears to provide an efficient alternative to signatures [12]. Rather than using asymmetric cryptography, TESLA uses symmetric cryptography with delayed key disclosure to provide the necessary asymmetry to prove the sender was the source of a message. Since symmetric cryptography is orders of magnitude faster than signatures, TESLA is resilient to computational DoS attacks. However, TESLA is vulnerable to memory-based Denial of Service attacks. In TESLA, receivers store data until the corresponding key is disclosed. Malicious parties can flood receivers with invalid messages which never have a corresponding key disclosure as part of a "pollution attack" [12]. If an attacker can fill a receiver's memory with junk data, performance on the receiver's system degrades. To address such memory-based DoS attacks in TESLA, we propose TESLA++ a modified version of TESLA that reduces memory requirements on the receiver without sacrificing security.

Alas, we cannot abandon digital signatures. At this time, VANET applications are still in the process of being defined, leaving their authentication requirements unclear. In addition, manufacturers may also develop new applications which require additional security properties which were previously considered unnecessary. Rather than proposing an authentication mechanism that focuses on one aspect (e.g., computation or bandwidth overhead, DoS resilience, or security requirement), we propose a flexible solution that provides a wide range of possible authentication properties and enables developers to fine tune parameters at a later time to achieve important properties.

The remainder of this work is organized as follows: Section 2 contains a summary of previous work on broadcast authentication. In Section 3, we discuss the different requirements for an authentication framework and why previous works fail to fulfill all of the prerequisites for a robust authentication framework. In Section 4, we introduce our DoS resistant version of TESLA, TESLA++. Section 5 contains the description of our authentication framework, VAST. In Section 6, we evaluate VAST through a series of simulations. In Section 7, we discuss some remaining topics which were not addressed earlier in the paper. We make concluding remarks in Section 8.

2. PREVIOUS WORK

Several works have investigated how to perform broadcast authentication [4,5,9,14,15] and how to mitigate Denial of Service (DoS) attacks against broadcast authentication [4,9,11].

Broadcast Authentication. To perform broadcast authentication, several works use asymmetric cryptography where the sender digitally signs messages or some structure which links messages together [4, 9, 15]. TESLA [14] and its derivatives use symmetric cryptography for broadcast authentication and rely on time to provide the necessary asymmetry so only the sender can generate a broadcast authenticator at a given time. Symmetric cryptography significantly reduces computation, but cannot provide non-repudiation (i.e., a recipient using TESLA cannot convince a third party that the sender indeed broadcast the message).

The IEEE 1609.2 VANET standard calls for the inclusion of an Elliptic Curve Digital Signature Algorithm (ECDSA) signature in every packet as a means for broadcast authentication [7]. Work by Raya et al. demonstrated that resource-constrained 400MHz machines intended for use in VANETs could handle the workload associated with asymmetric cryptography [15]. However, Raya's work assumes NTRU signatures which require less than $1/4$ of the time to verify. NTRU signatures are roughly 200 bytes (5 times the size of ECDSA signatures) and present significant overhead when included in every heartbeat message (a 32 byte or smaller message).

Researchers have proposed techniques which require less than one signature per packet as a means to reduce computation and bandwidth overhead associated with authentication. Broadcast Authentication Streams [4] and Distillation Codes [9] use error correction and limited digital signatures to address the scenario where a subset of a sender's packets are dropped or attackers inject malicious packets into the data stream. Using these techniques, a sender processes n packets as a set and only generates 1 signature for all n packets. Such processing prevents the sender from broadcasting any of the packets until the data in the last packet is known. This requirement introduces a delay, which is unacceptable in VANETs, since the sender will not know data for future heartbeat messages (i.e., the OBU's future location and velocity).

As an alternative to broadcast authentication based on asymmetric cryptography, TESLA [14] uses symmetric cryptography and delay key disclosure and time synchronization to provide the necessary asymmetry for broadcast authentication. In TESLA, a sender pre-computes a hash-chain of keys: $K_i = h(K_{i-1})$. The sender uses each of these keys for a short period of time to generate Message Authentication Codes (MACs). A certificate authority signs a copy of the hash chain anchor (K_n), the starting time for the hash chain, and the length of each key interval as a certificate for the sender. When a sender wants to broadcast a message M, the sender broadcasts M and the MAC of M generated with the key for that interval $K_i : \mathrm{MAC}_{K_i}(M)$. Once the time interval for K_i is over, the sender broadcasts K_i and starts using K_{i-1} to generate MACs for any messages broadcast in the new interval. Receivers store the message and the MAC until the key is broadcast. To authenticate a message, receivers hash the received key and compare it to the key in the certificate to verify the keys validity and use the now verified key to check that the stored MAC was generated with the appropriate key at the appropriate time. The maximum synchronization between senders and receivers controls the length of the time interval and subsequently the minimum authentication delay. Hu et al. propose the use of TESLA within VANETs [5] to reduce the overhead associated with authentication. As we discuss in Section 3 the fact that receivers must store messages provides a possible memory-based Denial of Service attack.

Denial of Service Mitigation. Several works have examined how to mitigate DoS attacks against broadcast authentication mechanisms. These schemes use puzzles [11] or filters [4, 9] to prevent receivers from expending resources on maliciously injected packets.

Ning et al. [11] propose the use of *message specific puzzles* to prevent DoS on broadcast authentication. Message-specific puzzles are computational puzzles [8] which force the sender to expend some amount of computation before receivers accept the message as legitimate. Parties can generate valid puzzle solutions at a rate proportional to the computation invested. This reduces the effectiveness of a computationally bounded attacker. However, the technique is inappropriate for VANETs where a sending OBU will have little spare computation power. Solving a new puzzle for each message introduces significant computation and delay at the sender.

Gunter et al.'s Broadcast Authentication Streams (BASs) [4] use forward error correction in broadcast streams such that the sender has to generate one signature for several packets. To mitigate DoS attacks where an attacker inserts invalid signatures, they propose selective verification where only a fraction of the signatures are verified. This approach is inappropriate for VANETs since a sender must know the contents of every packet in a set before the sender is able to compute the error correcting data which is inserted into each packet. Since an OBU lacks knowledge of the vehicle's future location and velocity this scheme would introduce an unacceptable delay as the OBU queued up packets in the set.

Karlof et al. propose the use of Distillation Codes [9] to prevent computational DoS in broadcast authentication where malicious parties inject spurious data in an attempt to interfere with error correction. This allows receivers to efficiently "distill" the sender's packets from malicious packets in the broadcast stream while permitting the sender to use one signature for a set of messages. Again, senders must process packets as sets. For scenarios where the sender knows data in advance this technique works well. As mentioned in the previous paragraph, the need to simultaneously process a set of packets introduces a delay which makes the technique inappropriate for VANETs.

This section has provided a description of previous work on broadcast authentication and ways to address DoS attacks against broadcast authentication. Next, we discuss the different properties VANETs require of a broadcast authentication mechanism and why the current solutions fail to meet all of these properties.

3. REQUIREMENTS AND COMPARISON OF BROADCAST AUTHENTICATION SCHEMES

In this section, we discuss the desirable properties of a broadcast authentication mechanism, potential attacks against those properties, and whether or not proposed broadcast authentication mechanisms fulfill those requirements.

3.1 Broadcast Authentication Properties

A successful authentication mechanism should fulfill several properties: secure authentication, non-repudiation, Denial of Service (DoS) resilience, and support for multi-hop communication. We now discuss each of these properties in turn.

Authentication. Authenticated data ensures receivers can verify that the message received was sent by the appropriate entity and that it has not been modified in transit. If an attacker can pose as another entity or modify another entity's packets without being detected, the mechanism fails to provide secure authentication. One attack against authentication is to pose as another entity and generate or modify a packet, or block a future packet to prevent authentication of the data. Such an attack is possible when, for example, an attacker modifies a series of packets from sender A which lack signatures. When A broadcasts the signature for the last few packets, the attacker could block the signature such that receivers will find authentication of the data or the modified data impossible.

Non-repudiation. Non-repudiation allows a receiver to prove to a third party that the sender is accountable for generating a message. If the broadcast mechanism lacks non-repudiation, a mali-

Scheme	Authentication	Non-Repudiation	DoS Prevention		Support for Multi-hop Comm.
			Computation	Memory	
ECDSA for Every Packet	√	√		√	√
ECDSA in 1 of n Packets					
TESLA	√		√		
VAST	√	√	√	√	√

Table 1: Comparison of the different properties broadcast authentication schemes fulfill

cious party can claim another party generated the message.[1] For example, in TESLA once the symmetric key used to generate a MAC is broadcast, any entity can use the disclosed key to generate a MAC for an arbitrary message. A malicious party could also fail to broadcast the necessary verification data that would hold them responsible for that message. For example, in schemes that use one signature for n packets, an attacker can broadcast spurious data and never broadcast the corresponding signature packet.

Denial of Service (DoS) Resistant. A mechanism should require little computational or memory resources such that other OBU operations may proceed unimpaired. Given the relatively expensive nature of digital signature verification ($\approx 7ms$ for ECDSA [15]), an attacker can launch a computational DoS by flooding a receiver with invalid signatures such that the receiver wastes processing power to verify the signatures. TESLA incurs little computational overhead, but requires entities to store messages and message-authentication-codes (MACs) until the corresponding symmetric key is broadcast. An attacker can broadcast a large number of invalid malicious messages such that receivers expend an excessive amount of memory resources as part of a "pollution attack" [12].

Multi-hop Authentication. Given the limited radio range of DSRC radios (reliable up to 300 meters) [7], a VANET authentication mechanism should enable parties outside of a sender's radio range to authenticate messages after an intermediate party has relayed the message. Such multi-hop authentication is crucial for applications that disseminate data over long distances or require extensive time and distance for drivers to respond. For example, knowledge of a closed or congested road is more useful miles away from the incident on the highway. Unless your vehicle is near an off-ramp, information about a traffic jam 300 meters ahead (e.g., just around the corner) is almost useless. Signatures allow multi-hop communication as a result of the non-repudiation property because any receiver can use the signer's public key to verify the signature. Multi-hop authentication is possible in TESLA, but one of two undesirable use cases must happen: receivers will forward data before having authenticated the message, or the sender must generate multiple MACs using different keys (i.e., keys for interval i, $i+1$, $i+2$ etc.) so receivers can authenticate a packet after an interval and forward the data and future key broadcasts from the sender to receivers further away who uses the other MACs and subsequent key broadcasts to authenticate the packet.

3.2 Comparison

We now compare previous proposals for VANET authentication with our new protocol (VAST) with respect to the aforementioned requirements. Table 1 contains a summary of this comparison.

IEEE 1609.2 [7] (the proposed standard) suggests the inclusion of an ECDSA signature in *every* packet to provide broadcast authentication. A digital signature ensures instant authentication with non-repudiation. However, the long verification time enables computational DoS attacks by flooding OBUs with bogus signatures.

The inclusion of a digital signature in a subset of the broadcast packets (i.e., after $n-1$ packets the n^{th} packet includes a signature over the last n messages) can help reduce bandwidth and computation overhead associated with security, but fails to fulfill the properties necessary for a VANET authentication scheme. As discussed earlier, attackers can block other senders' signatures to prevent authentication. Attackers could also fail to generate a signature–posing as though the packet was lost–to avoid non-repudiation. Expensive signature verification operations permit computational DoS where attackers broadcast a large number of invalid signatures. Storing packets until the signature arrives permits memory DoS since malicious parties can send numerous junk messages which victims store, expecting the broadcast of a signature. Given signature verification requires a subset or all n packets to successfully authenticate the data, multi-hop communication is inefficient. Rather than forwarding only the relevant packets, nodes must forward multiple packets, making the scheme inappropriate for multi-hop communication. Error correction codes can reduce the number of packets necessary for verification. However, error correction adds more data and introduces delay since the sender must know the data in the entire set before broadcasting the first packet.

TESLA may work as a VANET authentication mechanism with less computation and bandwidth demands. However, since TESLA uses symmetric cryptography non-repudiation is impossible. As discussed before, TESLA fails to support efficient multi-hop communication. If senders are limited to one MAC per packet, two unfortunate things can happen: a relayer forwards unauthenticated data or a relayer sends potentially incorrect – but authenticated – data as its own. If relaying entities forward messages and MACs before receiving the corresponding key, receivers more than one hop away from the sender will receive the data early enough that they can authenticate the data once the key is broadcast. However, an attacker could send invalid message/MAC pairs which relayers will forward since they have no way to tell if the information is authentic. This wastes bandwidth and storage since receivers should have dropped the invalid messages. If relaying nodes wait until the key is broadcast, the relaying node can verify the message is valid before retransmitting the data. However, the nodes must use their own TESLA credentials to retransmit data which may not necessarily be true, even though it was authenticated. For example, a sender can falsely claim debris is on the road and use TESLA to send an authenticated message about the fake debris. Once a node authenticates the message, the receiver will relay the message to other nodes and use his own TESLA values to authenticate the message. If the false debris notification results in legal actions, TESLA's lack of non-repudiation prevents the relaying node from proving to a third party he did not craft the lie, but received the fake message from the original sender. If the sender includes multiple MACs in the packet, each hop can authenticate the message before relaying it to nodes further away. Such an approach consumes a large amount of bandwidth; the additional MACs increase the size of the original packet. In addition, when a node relays a packet P, the relayer has to rebroadcast P and any subsequent key broadcasts from the sender to ensure recipients can verify the different MACs in P.

VAST uses a combination of TESLA++ (a modified version of TESLA, which is resilient to memory-based DoS attacks) and digital signatures to provide authentication, non-repudiation, DoS prevention, and multi-hop authentication. In Section 5, we provide a detailed description of our scheme and exactly how we achieve

[1] The scenario where an entity broadcasts its private asymmetric key to defeat non-repudiation is outside of the scope of this work.

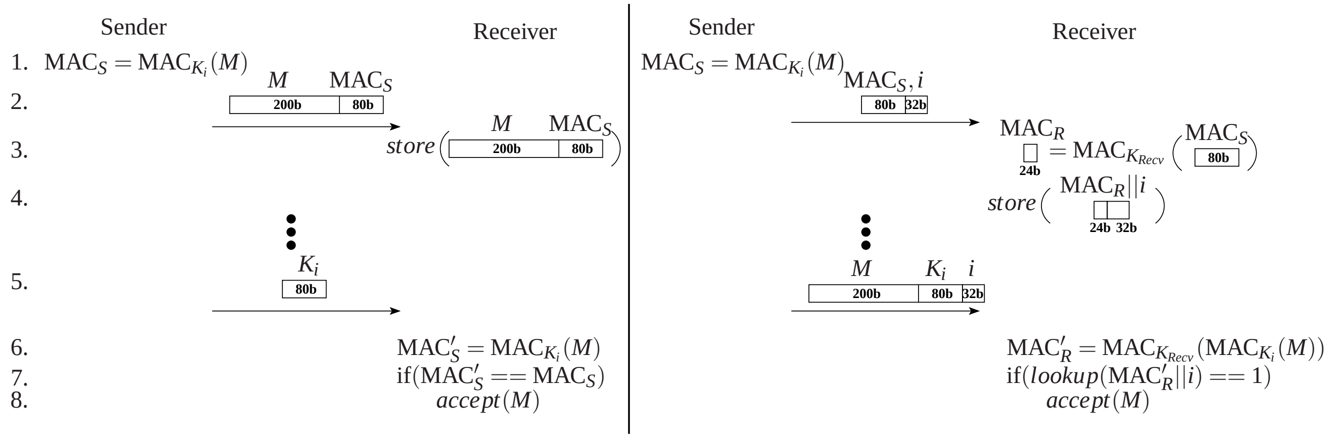

Figure 1: A comparison of TESLA and TESLA++

these properties. Before describing our entire scheme, we present TESLA++ and describe how it differs from TESLA in Section 4.

4. TESLA++

In this Section, we begin with a short description of TESLA [5,14] as background. We describe TESLA++ with an emphasis on how it improves on the techniques in TESLA. We also provide a security analysis of TESLA++ and a discussion of how TESLA++ provides resilience to memory-based DoS attacks.

Here we only present how a sender can perform broadcast authentication of a message within an interval. In TESLA and TESLA++, key management across intervals is the same (i.e., using key hash-chains) and any party wishing more information on that portion of the schemes should refer to the original TESLA publication [14].

TESLA Background. TESLA uses symmetric cryptography and delayed key disclosure to perform broadcast authentication (the left side of Figure 1 depicts the operations in TESLA). To authenticate a message M, a sender broadcasts the message and a Message Authentication Code (MAC) (Step 2) of the packet using the sender's key for this interval (K_i). Recipients save the entire message and MAC (Step 3) until the sender broadcasts the key. After the key disclosure period, the sender broadcasts the key (Step 5). To authenticate the message, receivers verify that the stored message/MAC pair agrees with the broadcast key (Steps 6 and 7). As we mentioned in Sections 2 & 3, one problem with TESLA is that receivers store all message/MAC pairs. With enough pairs maliciously broadcast, a pollution attack occurs where a receiver wastes a significant amount of memory storing invalid data [12].

TESLA++. We propose TESLA++ to prevent memory-based DoS attacks against TESLA. Like TESLA, TESLA++ provides broadcast authentication using symmetric cryptography and delayed key disclosure. However, in TESLA++, a receiver only stores a self-generated MAC to reduce memory requirements. Since receivers only store a shortened version of the sender's data, the sender first broadcasts the MAC and later broadcasts the corresponding key and message (similar to the Guy Fawkes protocol [1]). Figure 1 shows an example of how to authenticate a broadcast message using TESLA++.

To authenticate message M, in TESLA++, the sender first broadcasts the MAC ($\text{MAC}_S = \text{MAC}_{K_i}(M)$) which is computed with the current key K_i, along with the key index i (Step 2). Upon reception, using the key index i and the time associated with the start of the sender's key chain, a recipient first verifies the security condition to ensure that the key K_i for the sender has not yet been broadcast and is thus still only known by the sender. If the security condition does not hold, the receiver drops the MAC because an attacker could potentially have already received the corresponding key K_i. The receiver then re-MACs the received data using a local secret key K_{Recv} that is only known to the receiver ($\text{MAC}_R = \text{MAC}_{K_{Recv}}(\text{MAC}_S)$) (Step 3) and stores this shortened MAC (MAC_R) along with the key index (Step 4).

Once the key K_i can be disclosed, the sender will broadcast any messages and the key used to calculate the messages' MACs (Step 5). To verify a message, the receiver first verifies the validity of K_i by following the one-way key chain back to a trusted key. The receiver then calculates the shortened MAC of the message (Step 6) and compares it with the MAC and index stored in memory (Step 7). If the receiver has a matching MAC/key index pair in memory, the receiver considers the message authentic (Step 8). If none of the stored pairs match the newly calculated value, the receiver considers the message unauthentic and discards the message.

Over time the receiver will store more MAC and key index pairs in memory. When a stored MAC successfully authenticates a message, the receiver can free the memory used to store the MAC and key index. However, when the receiver misses a legitimate senders message and key broadcast or malicious nodes flood the network with MACs in an attempt to waste a receiver's resources, the receiver will need a policy to determine when to replace a MAC and key pair. In the event of a MAC flood and the receiver has insufficient memory, the replacement policy for shortened MACs stored in memory is an intricate issue in the design of TESLA++. For the replacement policy below, receivers also store the sender id and an arrival timestamp along with the shortened MAC and the key index (for simplicity, we left it out of the description above). For each sender (besides the trusted key chain value and key disclosure information), the receiver also stores the latest key index for which an authentic message has arrived. If memory space becomes insufficient, we make use of the following policy to identify which shortened MACs to discard:

- All shortened MACs with key indices that are older than the last authentic message received from that sender. The intuition is that older shortened MACs are still stored because an attacker injected the message or the corresponding message and disclosed key were lost.
- If more space is needed, the message whose verification is furthest out in the future is discarded. This addresses the scenario where attackers try to trick receivers into storing messages for a long period of time by claiming the key index is n when the real sender's current key index is j where $j << n$.

The DoS protection of TESLA++ comes at a cost: lack of non-repudiation, poor multi-hop performance, and poor functionality in lossy networks. Like TESLA, TESLA++ uses symmetric cryptography and as a result prevents computational DoS, but does not provide non-repudiation or efficient multi-hop authentication. In addition, senders using TESLA++ broadcast the MAC and the message in separate packets which impacts the functionality in lossy networks. In TESLA, the receiver acquires and stores the MAC and message together and can use any future key broadcast to authenticate the message. In TESLA++ if the appropriate message broadcast is lost the MAC is useless. We discuss the impact of this difference in VANETs later in Section 7. One solution is for the sender to broadcast the message with the MAC and allow the receiver the option of storing the message. A receiver that stores the message can use any future key broadcast to authenticate the message. Broadcasting the message multiple times presents a tradeoff between resilience to lossy channels and bandwidth overhead.

Storing smaller MACs and discarding old MACs makes TESLA++ resilient to pollution attacks [12].In the next subsection, we discuss why TESLA++ is secure and resilient to DoS attacks. However, TESLA++ fails to provide all of the properties necessary for a VANET authentication framework; TESLA++ lacks non-repudiation and multi-hop authentication. Without these we need the full authentication framework of VANET Authentication using Signatures and TESLA++ (VAST) to meet the VANET requirements defined in Section 3.

4.1 Analysis of TESLA++

This section analyzes the security and storage requirements of TESLA++. We begin by assuming TESLA and the underlying cryptographic functions (MACs and hashes) are secure. However, TESLA++ raises some questions since senders first broadcast a Message Authentication Code (MAC) and receivers generate a shorter MAC based on the received MAC and a secret key. Storing only the shortened MAC, instead of the original MAC and message, reduces the possibility of memory exhaustion attacks. However, if storing only a shortened MAC enables malicious parties to spoof other entities the technique is useles. In this section, we will discuss why broadcasting the MAC without the message is secure, why receivers can use shorter MACs when storing records of received MACs without decreasing security, and some rough calculations to demonstrate the memory savings and thus DoS resilience of TESLA++.

Attacks on Broadcasting MACs Alone. Under TESLA++, a sender first broadcasts the MAC and the key index and includes the message in the key broadcast. Some may worry that without the message and the MAC in the same packet, attackers can generate false messages and pose as the original sender. Provided secure underlying MACs and key hash chains, the probability of success for this attack is negligible. If an attacker waits until the key and message are broadcast, the attacker will try to find a different message which results in the same MAC as the original sender's message (i.e., find a new message M' such that the original message (M) and key (K_i) result in the same MACS ($\mathrm{MAC}_{K_i}(M) == \mathrm{MAC}_{K_i}(M')$)). Generation of such a message implies the underlying MAC was not CMA secure. An attacker can try to calculate the key before the original sender broadcasts the message and key. With knowledge of the key, the attacker can generate any valid MAC and message pair. For this attack to be successful, the sender must calculate the next TESLA++ key and generate a new MAC (or use the old one) such that the calculated key and desired message generate the broadcast MAC. To discover an undisclosed TESLA++ key, an attacker must defeat the one-way property of the hash used to build the hash chain, which is computationally infeasible. If an attacker broadcasts an arbitrary key (K') and message (which produce a previously broadcast MAC), a receiver can verify that K' is invalid by hashing the broadcast key (K') and comparing its value to previous keys from the claimed sender. Provided the underlying MAC algorithm and hash chain are secure broadcasting the MAC without the message in TESLA++ is secure.

Attacks on Storing Shortened MACs. In TESLA++, the receiver only records a shortened re-keyed MAC as a means to reduce storage. When receivers' keys are kept secret, TESLA++ provides security guarantees based on the size of the interval and the bandwidth of the medium. This is different and much easier to control than other cryptographic techniques which base security guarantees on computational capabilities which can vary greatly across attackers (e.g., a nation state versus a lone attacker with a laptop).

To take advantage of the shorter stored MAC, an attacker wants a smaller stored MAC to match the MAC for an attacker selected message using a legitimate party's key. For example if the shortened MAC is calculated as $\mathrm{MAC}_{K_{recv}}(X)$ where X is a broadcast MAC and an attacker wants to spoof a message M', the attacker will try to broadcast a MAC value Y such that after the spoofed sender broadcasts his/her key for the interval (K_i) the MAC for the attacker's message matches the receiver's stored MAC (i.e., $\mathrm{MAC}_{K_{recv}}(\mathrm{MAC}_{K_i}(M')) = \mathrm{MAC}_{K_{recv}}(Y)$). With more stored MACs, the chance that a message key combination (and corresponding MAC) corresponds to a previously heard MAC increases. However, the receiver's key (K_{recv}) is secret so an attacker cannot calculate the shortened MAC for a given broadcast value.

Without knowledge of the receiver's key, an attacker's best strategy is to broadcast as many MACs for a given key interval as possible in an attempt to make it appear as though an attacker generated message and a legitimate user's key correspond to a previously heard MAC. If a receiver believes it has heard every possible MAC in the appropriate key interval, the receiver will mistakenly verify every TESLA++ key and message pair it receives as authentic since it will have a record of the corresponding MAC. Assuming the re-MACing process uniformly assigns MACs, this problem reduces to the coupon collector problem where each attacker broadcast MAC is an attempt to have a receiver record a new shortened MACs.

Even with a very short stored MAC, an attacker will have a difficult time fooling a receiver with an arbitrary message. With a relatively short stored MAC of 16bits, there are $2^{16} \approx 64000$ shortened MACs and the attacker needs to send on average $2^{16}\log 2^{16} = 2^{20}$ or roughly one million MACs to ensure he can forge an arbitrary message from a sender in a key interval. In the case of VANETs with a DSRC bandwidth of 56Mb/s, a 100ms TESLA++ interval, and an 80bit sender MAC, an attacker can only send ≈ 70 thousand MACs in an interval. As such, the probability of an attacker successfully fooling a receiver with an arbitrary message with a 16bit stored MAC and the aforementioned bandwidth and interval is around 7%. A 32 bit MAC would reduce the probability of success to 10^{-6}. If we consider the additional overhead for each packet's header and the key index, the actual number is smaller. When attackers cannot find collisions in the larger broadcast MAC, TESLA++ with small time intervals and relatively small receiver MACs provides a negligible probability that an attacker can spoof another sender as a result of the storage optimizations, independent of the computational power of the attacker(s).

Maximum Storage. In the previous paragraph, we showed how TESLA++ remains secure even when storing smaller MACs. The reason to use smaller MACs is to reduce storage constraints in TESLA++ and prevent pollution/memory-based DoS attacks. Here we discuss the upper-limit on memory consumption for TESLA++ in different VANET configurations. When storing only re-MACed values the maximum memory consumption is a function of the maximum number of MACs which can be broadcast in an interval and how long MACs are stored. Given, the acceptable latency is on the order of a few hundred milliseconds in VANETs [3], the TESLA interval should be made small (50 to 100ms) to ensure messages are quickly authenticated. This also implies that senders should broadcast messages within the next couple of intervals. If a

Step	Sender		Receiver
1.	$\sigma_{Si} = Sign(M, K_{Sender}^{-1})$		
2.	$\mu_{Si} = \text{MAC}_{K_{Send_i}}(M\|\|\sigma_{Si})$		
3.		$\xrightarrow{\mu_{Si}, i}$	
4.			$\mu_R = \text{MAC}_{K_{Recv}}(\mu_{Si})$
5.			$store(\mu_R\|\|i)$
	$\vdots$	(a TESLA++ interval later)	
6.		$\xrightarrow{M, \sigma_{Si}, K_{Send_i}}$	
7.			if$(h(K_{Send_i})! = K_{Send_{i-1}})$
			ignore(M)
			else
8.			$\mu'_R = \text{MAC}_{K_{Recv}}(\text{MAC}_{K_{Send_i}}(M\|\|\sigma_{Si}))$
9.			$if(lookup(\mu'_R\|\|i) == 1)$
10.			$Verify(\sigma_{Si}, M, K_{Sender}^{+})^*$
11.			$accept(M)$
			else
12.			$if(CPU\_util < \omega$ and $\#\_Mess\_In\_Queue < \lambda)$
13.			$Verify(\sigma_{Si}, M, K_{Sender}^{+})$
14.			$accept(M)$

Figure 2: VANET Authentication using Signatures and TESLA++ . K_{Send_j} are symmetric keys used for TESLA++. $K_{Sender}^{+/-1}$ are ECDSA keys. $*$ Step 10 is only performed when non-repudiation is necessary.

MAC has a key index that corresponds to disclosure multiple intervals in the future, receivers can ignore the MAC since the data will be old by the time message and key are broadcast. The real time requirement in VANETs reduces the maximum number of MACs stored to less than the maximum number that could be broadcast in two TESLA++ intervals ($\leq$ 200ms). Given VANETs have a bandwidth of $56Mb/s$ [7], an OBU will have to store at most the maximum number of bits transmitted in 200ms times the space savings of the receivers MAC, or $11.2Mb \cdot \frac{|MAC_{recv}|}{|MAC_{send}|}$ [2]. For example, if broadcast MACs are 80 bits and receiver MACs are 24bits long, receivers only have to reserve less than 1/2 a megabyte of space. Even with a limited space of 1 megabyte, a receiver can handle more than the maximum amount of data an attacker can force the receiver to store. In this section we have described a modified version of TESLA, TESLA++, which reduces the storage requirements for receivers without reducing security. As such, TESLA++ provides a broadcast authentication scheme based on symmetric cryptography without a vulnerability to memory-based DoS attacks.

5. VANET Authentication using Signatures and TESLA++ (VAST)

VANETs require an authentication framework which provides more than just authentication of packets. Non-repudiation is necessary for attribution and efficient multi-hop communication. The framework must also provide efficient and timely authentication to prevent flooding or computational DoS attacks. The previous works discussed in Section 2 and Section 3 were good first approaches to VANET authentication, but are not flexible enough to meet all of the properties discussed in Section 3. In this work we propose a new framework, VANET Authentication using Signatures and TESLA++ (VAST), which uses a combination of ECDSA signatures and TESLA++ to verify each packet. TESLA++ provides an efficient DoS resilient authentication mechanism to verify legitimate packets and filters out the majority of malicious or spurious messages. Once an OBU verifies the packet using TESLA++, the OBU may verify the ECDSA signature if non-repudiation is necessary (e.g., the message will cause a driver alert or any other situation where the message may negatively impact the driving experience). The signature also enables authentication for multi-hop communication. If the OBU has no record of the TESLA++ MAC, the OBU will verify the signature, provided the OBU's CPU and message buffer indicate it has processing power to spare. In this section, we present VAST and discuss how it meets the requirements set out in Section 3: authentication, non-repudiation, DoS resistance, and efficient multi-hop communication.

VAST is shown in Figure 2 where the sender broadcasts an authenticated message M. Note that receivers perform two types of verification: 1) a TESLA++ verification in steps 7, 8, and 9 and 2) digital signature verification in step 10 when the application requires non-repudiation or step 13 when TESLA++ authentication fails (possibly due to a lost MAC) and if CPU utilization and the number of messages in the processing queue are below certain thresholds (i.e., computational DoS is not an issue). These thresholds provide flexibility within VAST such that VANET system designers can mold the authentication framework to meet application needs. As such, the exact values of the thresholds depend on the suite of VANET applications and should be selected once the application requirements are defined.

TESLA++ provides authentication and a filter of the data broadcast during times of high computational load. The previously received and recorded MAC (steps 2 to 5) ensures the validity of the message and the signature while the hash chain ensures the proper key is used (step 7). The digital signature included with every message provides non-repudiation in case the relevant application requires non-repudiation or M must be forwarded to other VANET participants which may have missed the broadcast of the original TESLA++ MAC (step 3).

Under VAST, the digital signature is authenticated using TESLA++ (steps 7 to 9) before it is verified, preventing the majority of computational and memory-based DoS attacks. Authenticated signatures prevent attackers from broadcasting invalid signatures while posing as other VANET entities. In the case where the receiver

[2] Note that the per broadcast packet overhead of source address and lower layer information overshadows the receiver stored key index and other data used to determine when to replace a MAC.

has no record of the TESLA++ MAC, the receiver will only verify the signature if the extra computation will not lead to a DoS (see step 12). We choose to use CPU utilization (ω) and number of messages in the processing queue (λ) to determine thresholds for acceptable computational load, but other metrics could be used. The only way a malicious party can trick receivers into verifying digital signatures during times of high computation is by sending a TESLA++ authenticated signature. Under such a scenario, recipients can determine which entity sent the signature, and ignore signatures from any sender that has a history of broadcasting invalid signatures. The storage techniques used in TESLA++ (see Section 4 and steps 4 and 5 in Figure 2) reduce storage needs and prevent pollution attacks [12].

VAST allows for multi-hop communication and authentication through the use of both TESLA++ and ECDSA signatures. Vehicles further away will miss the sender's original TESLA++ MAC broadcast so ECDSA signatures are needed for authentication. However, if OBUs were to simply verify any signature they receive, a computational DoS attack would be possible. Instead, the relaying OBU should include the original sender's/forwarded message and signature ($M_{fwd}||\sigma_{fwd}$) as part of the relaying OBU's own messages ($M_{relay} = M_{new}||M_{fwd}||\sigma_{fwd}$) which are authenticated using either the relaying nodes signature or TESLA++ authenticator. Now, the recipient several hops away can use TESLA++ to verify the validity of the relayers message (which includes the original sender's signature) and only if that is authentic will the recipient expend the computation to verify the original sender's signature in the forwarded message. In the case where the TESLA++ data allows authentication, but the forwarded signature is invalid, the receiving OBU can label the relaying OBU as a potential attacker and ignore the relaying OBU's future messages. In the case with authentic, but false data in the original message (i.e., the sender signed a lie), the signature in the original message indicates the true origin of the false data.

In this section, we presented VAST and explained how it fulfills the different requirements from Section 3: authentication, non-repudiation, DoS resilience (computation- and memory-based), and multi-hop communication. We discuss the simulation of ECDSA, TESLA, and VAST in Section 6 and compare the performance of each.

6. SIMULATION OF AUTHENTICATION MECHANISMS

To evaluate the efficacy of our scheme, we use ns-2 [16] to simulate VANETs using ECDSA, TESLA, or VAST on a 1 kilometer long stretch of a large highway (4 lanes of traffic in each direction with 50 meter median between each side of the highway) with varying traffic densities, traffic speeds, and packet error rates. We only simulate highway traffic since this presents a scenario where the authentication framework encounters the greatest load due to a large number of vehicles within range at a given time. During simulation each vehicle broadcasts a heartbeat message every 100ms [2]. This heartbeat message contains the size of the packet, the OBU's address, location, and velocity, the broadcast address (as the receiver address), and the authentication data as contained in Table 2. For simulation, the OBU's radio range is set to 300m, signal attenuation is modeled according to ns-2's two ray ground model, and the bandwidth is one DSRC channel (6Mb/s) [7]. For this simulation we focus on the overhead associated with message authentication and ignore the certificate broadcast and verification process since it is the same for each mechanism (i.e., only one signature from an authority is necessary to verify a sender's public key, TESLA anchor, or public key and TESLA++ anchor).

For simulation we assume OBUs' cryptographic performance corresponded to the values from Raya et al. [15] shown in Table 3. To analyze the performance of the different schemes under different traffic scenarios we use the different values summarized in Table 4. For simulation of ECDSA, we assume a fixed size queue to store up to 50 messages while waiting for signature verification and that if the queue was full any received message was dropped. A larger queue would decrease the number of dropped packets, but would also increase authentication delays since packets would be in the queue longer. For simulation of TESLA, we consider any message that was not verified within 1 second as dropped. For simulation of VAST, we assume that if the message queue is larger than 10 messages ($\lambda = 10$) the message is dropped. For our simulation, we allow full CPU utilization ($\omega = 100\%$) since the number of messages in the queue provides sufficient evidence of computational DoS (i.e., if the message queue is growing the OBU is receiving messages faster than it can process them).

In each traffic scenario, OBUs drive for 1 minute of simulation time to fill their queues and begin to process messages. After this warm-up period, we simulate the VANET for an additional 10 minutes of simulated time where each OBU in the 1km stretch of the highway records the total number of messages received, number of messages dropped (due to full processing queues or long time between message reception and key broadcast), and authentication delay. Authentication delay is defined as the amount of time between when the sending OBU knows the data and when a receiver can authenticate the data. In our simulation we choose to have TESLA++ and TESLA piggyback future MACs or key exposures in the current heartbeat message. This optimization reduces bandwidth usage since key exposure can occur in the same message as a future MAC, but as a result the smallest possible authentication delay for those schemes is the time between two heartbeat messages (100ms).

Structure	Size
Vehicle Info	192 bits
ECDSA Signature	320 bits
MAC, MAC KEY	80 bits
ECDSA Only Packet Contents	64B
TESLA Only Packet Contents	44B
VAST Packet Contents	84B

Table 2: Size of Data in the Various Packets

Operation	Comp. Time
ECDSA generation	4ms
ECDSA verification	7ms
Symmetric Cryptography (Hash or MAC)	1μs

Table 3: Computational Times of Simulated Cryptographic Operations

6.1 Simulation Results

Figures 3 and 4 show the impact of increasing traffic density on the percentage of received packets processed (i.e., 1 - percent dropped) and the average authentication delay. For these scenarios the average vehicle speed was fixed at 30m/s (70mph) and 10% of packets were uniformly dropped at random due to wireless reception errors. Across all scenarios, VAST performs well with little au-

Quantity	Range
Traffic Density	1 - 75 cars in radio range
Wireless Errors	P(error) = 0.00 - 0.50
Traffic Speed	10m/s (20mph) - 40m/s (90mph)

Table 4: Simulated Traffic Values

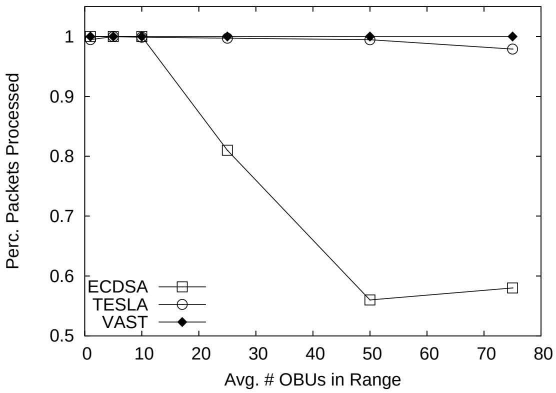

Figure 3: Packet Process Rate vs. Traffic Density

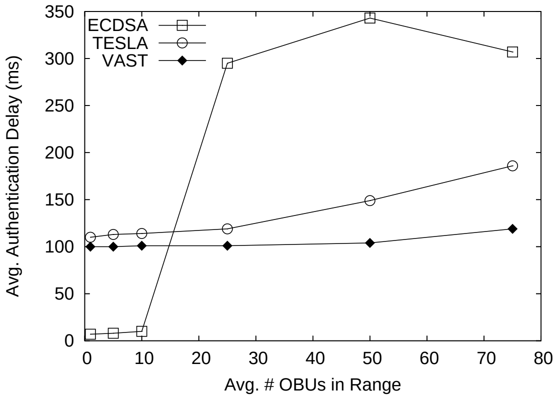

Figure 4: Authentication Delay vs. Traffic Density

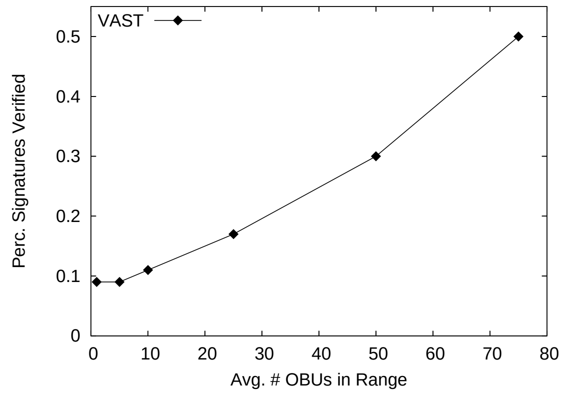

Figure 5: Portion of Signatures Verified vs. Traffic Density

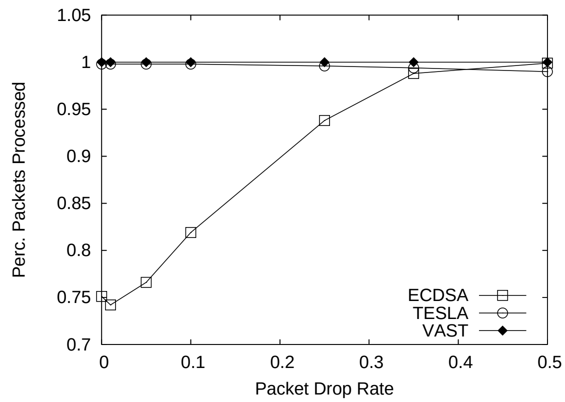

Figure 6: Packet Process Rate vs. Wireless Losses

thentication delay and 100% of data received authenticated. As traffic density increases, when OBUs only use ECDSA the processing time is too large and as queues fill up delays increase and packets are dropped. For OBUs using TESLA, denser traffic introduces delays when channel contention causes more messages to be missed. For VAST, as traffic density increases and more packets are missed due to channel contention, OBUs use signatures to verify packets when the corresponding MAC was missed. However, OBUs used TESLA++ to authenticate a significant fraction of packets so processing queues remain relatively empty. Figure 5 shows the percentage of received packets that were authenticated using signatures under VAST and confirms that as more packets are lost due to contention VAST utilizes the included signatures to authenticate the messages. When 75 OBUs were in range, channel contention reduces the number of received packets such that 50% of those packets is less than the number of packets received during the 25 cars in range scenario, allowing for VAST to handle that many signature verifications. This finding indicates that the channel, rather than OBU processing capabilities, limits the rate of data authentication possible in our simulation of VAST.

Figures 6 and 7 show the impact of increasing losses in the wireless network on packet processing capabilities and authentication delay. The vehicles' speed was fixed at 30m/s (70mph) and traffic density was 25 cars in radio range. VAST performs well independent of the error rate as it smoothly adjusts to different error rates, using TESLA++ the majority of the time when error rates are low and using more signatures as error rates increase (see Figure 8). When packet error rates are low, VAST uses TESLA++ to avoid excessive computation. With more packets lost to wireless errors, VAST begins verifying signatures in packets since the corresponding MACs were lost. ECDSA performs well with more wireless errors. With more errors there are less packets received. This reduces the computational load due to signature verification and improves packet processing rate compared to previous simulations. The increase in packet loss increases authentication delay for TESLA since it is several intervals between when an OBU receives a message and a MAC and when the OBU receives a key it can use to verify the MAC. As a result, when approximately every other packet is dropped (50% drop rate) the authentication delay increases to approximately two intervals.

We also ran simulations with speeds varying between 10m/s (20mph) and 40m/s (90mph), but the change in speed did not have a statistically significant impact on packet processing capabilities or authentication delays.

The simulation results in this section show that our scheme is flexi-

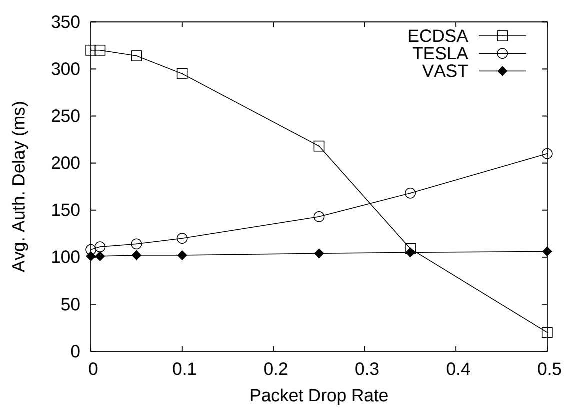

Figure 7: Authentication Delay vs. Wireless Losses

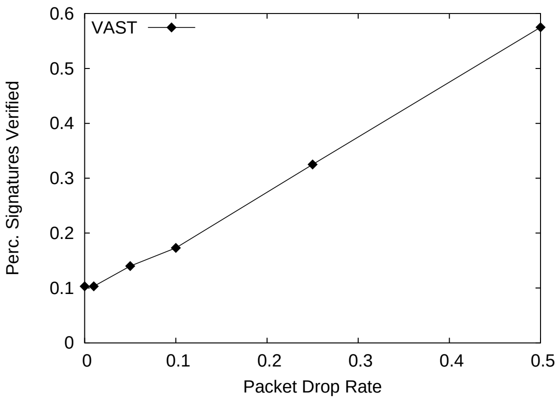

Figure 8: Portion of Signatures Verified vs. Loss Rate

ble and efficient enough to provide timely authentication of VANET messages under a wide range of scenarios that produce ill effects for prior VANET authentication mechanisms.

7. DISCUSSION

In this section we discuss some remaining issues which were not addressed earlier in the paper.

Authentication Delay. The delay between when a node receives a message and when the node can authenticate the message is an important value in VANETs. For example, safety messages require a small authentication delay, otherwise drivers will not have sufficient time to respond to an alert in a dangerous situation.

For VAST, the authentication delay depends on the time between when the sender broadcasts the MAC and when the sender broadcasts the message, the key, and the signature. In both cases, the delay is roughly the time between when the sender knows the data and when the sender reveals the data. The sender could reveal the data and a signature along with the MAC (before revealing the TESLA++ key), but OBUs should rely on TESLA++ for authentication to prevent computational DoS due to signature verification. As a result, the receivers will wait until at least the TESLA++ key is broadcast (or should have been broadcast) before the message is authenticated.

TESLA++ can utilize the parameters defined for TESLA and achieve a similar authentication delay. According to Perrig et al. [13] the delay between MAC and key broadcasts within TESLA is a function of the maximum synchronization error between nodes, the maximum network delay between hosts, and the size of the TESLA time interval. If GPS synchronization is used, synchronization between nodes is around 20*ns* for expensive GPS units [10] and less than 100*ns* for more economic devices. Given the 1*km* maximum transmission range of DSRC and a 6 Mb/s throughput (the rate for a single channel of DSRC), the network delay is less than 5*ms* for single-hop communication. If OBUs will use a given key chain for only a few minutes to help achieve privacy [15], we can select a shorter interval to help reduce authentication delay. If we assign an interval of 5*ms*, the authentication delay associated with TESLA++ is 1 interval or 5*ms*. However, this prevents senders from piggybacking messages like we did in simulation (where the MAC for the next message was included in the current heartbeat message). If senders use an additional broadcast, specifically for key disclosure, each sender must broadcast not only the key, but also all of the lower level data (MAC and Physical layers) associated with a packet for an extra 40bytes of broadcast information [6]. The sender only broadcasts a key at the end of the interval after every message (as opposed to at the end of every interval) so the bandwidth usage is a function of the number of messages. Without more real world data about acceptable authentication delays and effective throughput of the DSRC channel it is difficult to make any definite statements about which approach is better: a small interval with more packets or a larger interval which corresponds to the heartbeat message interval.

Packet Loss. For TESLA++ to work successfully a receiver needs both the original MAC packet and the packet with the message and the key. If the receiver misses the message and key packet (only the MAC is received), the receiver will not have the data. A similar issue is present in TESLA when the recipient only receives the key broadcast they will not know what data that key authenticates. When only messages are received, but the MAC for VAST was lost or the next key broadcast for TESLA is lost authentication is still possible. In VAST, the receiver can use the signature to verify a message if TESLA++ authentication fails and the processing queue is not full. In TESLA, a receiver can use any future key to authenticate a previously received packet.

Even if receivers do miss a small number of heartbeat messages applications will still work. The VANET heartbeat messages used for most safety applications are frequently broadcast (every 100ms) and each message overrides the values from previous messages (i.e., the vehicle's current position and velocity is more important than where it was a few moments ago) [2]. As such, even if a VANET recipient misses a message and key packet, the sender will broadcast updated location and velocity information within a relatively short period of time. Bai et al. [2] discuss this issue using their terms of "network-level metrics" and "application-level metrics". The probability of packet loss is a network-level metric for reliability. While, some applications only need one message within a given time window to work ("Application-level T-Window Reliability"). Even with poor network reliability, application reliability is fairly good. For example, if network reliability is 50%, an application with a time window of 0.5 seconds has a reliability of 97%.

8. CONCLUSION

In this paper, we analyze the different requirements of Vehicular Ad Hoc Network (VANET) authentication mechanisms and find that prior approaches fail to meet all of the necessary properties. To address this problem we propose a new authentication building block TESLA++ that represents a DoS resilient version of TESLA. Our authentication framework VANET Authentication using Signatures and TESLA++ (VAST) uses both ECDSA signatures and TESLA++ to provide timely and efficient authentication of VANET messages while remaining resilient to DoS attacks. Simulation results show that under a range of scenarios VAST authenticates 100% of the received data while maintaining acceptable authentication delays (worst case of 107ms). The combination of VAST and our certificate management techniques provide a complete system to efficiently manage authentication of VANET messages without exposing VANET participants to Denial of Service attacks.

9. REFERENCES

[1] R. J. Anderson, F. Bergadano, B. Crispo, J.-H. Lee, C. Manifavas, and R. M. Needham. A new family of authentication protocols. *ACM Operating Systems Review*, Oct. 1998.

[2] F. Bai, T. Elbatt, G. Hollan, H. Krishnan, and V. Sadekar. Towards characterizing and classifying communication-based automotive applications from a wireless networking perspective. In *Proceedings of IEEE Workshop on Automotive Networking and Applications (AutoNet)*, Dec. 2006.

[3] F. Bai and H. Krishnan. Reliability analysis of DSRC wireless communication for vehicle safety applications. In *Proceedings of IEEE Intelligent Transportation Systems Conference (ITSC)*, 2006.

[4] C. A. Gunter, S. Khanna, K. Tan, and S. S. Venkatesh. Dos protection for reliably authenticated broadcast. In *Proceedings of Network and Distributed System Security Symposium (NDSS 2004)*, Feb. 2004.

[5] Y.-C. Hu and K. P. Laberteaux. Strong VANET security on a budget. In *Proceedings of Workshop on Embedded Security in Cars (ESCAR)*, 2006.

[6] IEEE. Wireless LAN medium access control (MAC) and physical layer (PHY) specifications. IEEE Standards, 1999.

[7] IEEE. 1609.2: Trial-use standard for wireless access in vehicular environments-security services for applications and management messages. IEEE Standards, 2006.

[8] A. Juels and J. Brainard. Client puzzles: A cryptographic countermeasure against connection depletion attacks. In *Proceedings of the Symposium on Network and Distributed Systems Security (NDSS '99)*, Feb. 1999.

[9] C. Karlof, N. Sastry, Y. Li, A. Perrig, and J. D. Tygar. Distillation codes and applications to dos resistant multicast authentication. In *Proceedings of Network and Distributed System Security Symposium (NDSS 2004)*, Feb. 2004.

[10] M. A. Lombardi, L. M. Nelson, and A. N. Novick. Time and frequency measurements using the global positioning system. *Cal Lab: The International Journal of Metrology*, July–September 2001.

[11] P. Ning, A. Liu, and W. Du. Mitigating DoS attacks against broadcast authentication in wireless sensor networks. *ACM Transactions on Sensor Networks (TOSN)*, 2007.

[12] A. Perrig, R. Canetti, D. Song, and D. Tygar. Efficient and secure source authentication for multicast. In *Proceedings of the Symposium on Network and Distributed Systems Security (NDSS 2001)*, Feb. 2001.

[13] A. Perrig, R. Canetti, D. Tygar, and D. Song. Efficient authentication and signature of multicast streams over lossy channels. In *Proceedings of the IEEE Symposium on Research in Security and Privacy*, May 2000.

[14] A. Perrig, R. Canetti, J. D. Tygar, and D. Song. The TESLA broadcast authentication protocol. *RSA CryptoBytes*, 2002.

[15] M. Raya and J.-P. Hubaux. The security of vehicular ad hoc networks. In *Proceedings of the ACM Workshop on Security of Ad Hoc and Sensor Networks (SASN)*, Nov. 2005.

[16] VINT Project, University of Berkeley/LBNL. NS-2:network simulator. `http://www.isi.edu/nsnam/ns/`.