Flooding-Resilient Broadcast Authentication for VANETs [*]

Hsu-Chun Hsiao, Ahren Studer,
Chen Chen, Adrian Perrig
CyLab/Carnegie Mellon University
{hsuchunh, astuder, chenche1,
adrian}@ece.cmu.edu

Fan Bai, Bhargav Bellur, Aravind Iyer
General Motors Research
{fan.bai, bhargav.bellur,
aravind.iyer}@gm.com

ABSTRACT

Digital signatures are one of the fundamental security primitives in Vehicular Ad-Hoc Networks (VANETs) because they provide authenticity and non-repudiation in broadcast communication. However, the current broadcast authentication standard in VANETs is vulnerable to signature flooding: excessive signature verification requests that exhaust the computational resources of victims. In this paper, we propose two efficient broadcast authentication schemes, Fast Authentication (FastAuth) and Selective Authentication (SelAuth), as two countermeasures to signature flooding. FastAuth secures periodic single-hop beacon messages. By exploiting the sender's ability to predict its own future beacons, FastAuth enables 50 times faster verification than previous mechanisms using the Elliptic Curve Digital Signature Algorithm. SelAuth secures multi-hop applications in which a bogus signature may spread out quickly and impact a significant number of vehicles. SelAuth provides fast isolation of malicious senders, even under a dynamic topology, while consuming only 15%–30% of the computational resources compared to other schemes. We provide both analytical and experimental evaluations based on real traffic traces and NS-2 simulations. With the near-term deployment plans of VANET on all vehicles, our approaches can make VANETs practical.

Categories and Subject Descriptors

C.2.0 [**General**]: Security and protection; C.2.1 [**Network Architecture and Design**]: Wireless communication

General Terms

Algorithms, Design, Security

Keywords

Broadcast authentication, flooding resilience, signatures, VANETs

1. INTRODUCTION

Vehicular Ad-Hoc Networks (VANETs) enable inter-vehicle communication to improve road safety and driving experience. For example, in the Cooperative Collision Warning (CCW) application, vehicles exchange their location and speed to avoid accidents through periodic beacon messages. For efficiency, the Congested Road Notification (CRN) application enables vehicles in a traffic jam to alert approaching vehicles of the congestion [2]. VANET applications rely on vehicles' On-Board Units (OBUs) to broadcast outgoing messages and validate incoming messages. An OBU should be able to verify a message before the vehicle needs to act upon it (i.e., before the message's deadline).

One of the biggest concerns for VANET deployment is security. An attacker can not only cause financial loss but also threaten drivers' lives. The recent successes in attacking vehicular systems [8, 20, 35] have demonstrated the need to design vehicular networks with security in mind. An indispensable security primitive in securing vehicular networks is *broadcast authentication*, because it ensures that vehicles only accept messages from legitimate senders and the messages are unaltered during transmission. In addition to authentication, *non-repudiation* is required to prevent vehicles from denying the creation of sent messages and to enable the report of malicious participants to the legal authorities. Non-repudiation is especially important for resolving insurance cases after accidents happened. Though the IEEE 1609.2 standard [15] achieves both broadcast authentication and non-repudiation using the Elliptic Curve Digital Signature Algorithm (ECDSA), verifying every signature using ECDSA causes high computational overhead on the standard OBU hardware, which has limited resources due to manufacturers' cost constraints. A typical OBU with a 400MHz processor requires 20 milliseconds to verify one ECDSA signature.

Consequently, VANET authentication is vulnerable to **signature flooding**, where a vehicle receiving many signed messages in a short time period is unable to verify all of the messages before their respective deadline given the vehicle's constrained computational power. Despite the significant impact on availability, signature flooding can be triggered easily even without any malice. For example, a vehicle having more than five vehicles within radio range would be overwhelmed by its neighbors' beacon messages, sent 10 times per second as dictated by the current standard [15].

[*]This research was supported in part by CyLab at Carnegie Mellon under grants DAAD19-02-1-0389 and W911NF-09-1-0273, and MURI W 911 NF 0710287 from the Army Research Office, grants CNS-1040801 and CNS-1050224 from the National Science Foundation, and by support from General Motors through the GM-CMU Collaborative Research Laboratory. The views and conclusions contained here are those of the authors and should not be interpreted as necessarily representing the official policies or endorsements, either express or implied, of ARO, CMU, GM, NSF, or the U.S. Government or any of its agencies.

MobiCom'11, September 19–23, 2011, Las Vegas, Nevada, USA.
Copyright 2011 ACM 978-1-4503-0492-4/11/09 ...$10.00.

Such a flooding problem is exacerbated under the presence of attackers and/or multi-hop communication.

In this work, we observe that signature flooding can be mitigated by broadcast authentication schemes whose overheads match the *entropy* of the broadcast messages. We study this approach in the context of VANETs and propose two flooding-resilient broadcast authentication schemes, FastAuth and SelAuth, for different VANET applications. Our schemes are based on digital signatures thus providing non-repudiation. To the best of our knowledge, prior work on lightweight broadcast authentication either lacks the non-repudiation property or fails to operate efficiently in dynamic VANET environments. We briefly summarize our schemes.

Fast Authentication (FastAuth) enables real-time, lightweight authentication by exploiting the predictability of future beacon messages. In FastAuth, the lower the entropy of a future beacon (from the sender's point of view), the smaller the beacon message is. We design a new structure called *chained Huffman hash trees*, which supports a one-time signature scheme whose verification is 50 times faster and whose generation is 20 times faster than ECDSA. Also, FastAuth reduces the communication overhead to half.

Selective Authentication (SelAuth) provides fast isolation of malicious senders. As a result, invalid signatures are contained to a small area, without impacting the rest of the network. SelAuth enjoys a short convergence time because vehicles intelligently select which signed messages to verify before forwarding. This selection is performed in a way that signatures with an unknown state (low certainty on the validity, or high "entropy") are more likely to be checked. Specifically, we propose *warning pushback* to share information about flooding vehicles, and *forwarder identification* to distinguish benign neighbors from misbehaving ones. The evaluation shows SelAuth imposes 10%–35% computational overhead compared to other closely related schemes while containing 99% of invalid signatures to one hop.

Contributions. The main contributions of this work are:

- We identify that the current broadcast authentication standard for VANETs fails to operate under signature flooding, which occurs frequently even in benign settings.
- We propose two novel flooding-resilient schemes enabling timely and efficient signature verification; FastAuth leverages beacon predictability to secure safety beacons, and SelAuth utilizes neighbor information to secure multi-hop messages.
- Analytical and experimental results show that VANETs can be made practical with our schemes, which provide substantial performance advantages over prior work.

Organization. Section 2 provides background on VANET settings and cryptographic primitives. Section 3 presents the problem definition and the threat model. We introduce FastAuth and SelAuth schemes in Sections 4 and 5, respectively. We then evaluate FastAuth and SelAuth in Sections 6 and 7. Section 8 summarizes the prior work on broadcast authentication. We conclude in Section 9.

2. BACKGROUND

Before describing our proposed broadcast authentication schemes, we give an overview of the VANET setting as dictated by the IEEE 1609.2 standard [15] and provide background on basic cryptographic primitives we use.

2.1 VANET Standard

Public key infrastructure. As proposed by the IEEE 1609.2 standard, VANETs are required to have a Public Key Infrastructure (PKI) for key management. Each vehicle has a pair of ECDSA keys: a private signing key and a public verification key. The verification key is certified by a certificate authority (CA). Auto manufacturers or regional Department of Transportation agencies can act as CAs. Each key pair will then be stored in the vehicle's OBU, with tamper-resistant protection to protect the private key from compromise. To protect privacy and prevent location tracking, a VANET-enabled vehicle can obtain multiple certified key pairs with non-overlapping periods of validity and change its public key periodically (e.g., every five minutes) [32].

VANET message format. A VANET message contains a timestamped message body m, the sender's signature $S(m)$ on m, and the sender's public key certificate.

$$\langle m, S(m), cert \rangle \tag{1}$$

The timestamp states the creation time of this message, which helps the receiver determine this message's deadline. A vehicle should act upon a received message before the message's deadline. The signature ensures that the sender is accountable for sending this message, and thus deters selfish drivers from broadcasting bogus information for self-interest.

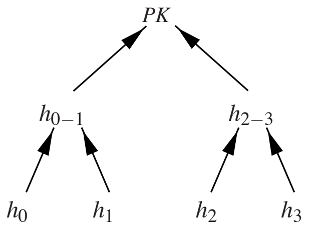
Application types. VANET applications can be classified into two types based on how frequent messages are sent and the distance these messages need to be propagated [12]. In *single-hop relevant applications*, vehicles periodically exchange *beacons* with nearby vehicles that are one hop away from each other. Beacons contain the sender vehicle's kinematics information such as position and velocity to enable timely reaction to unsafe conditions. The IEEE 1609.2 standard instructs vehicles to broadcast beacons 10 times per second. Message information is obtained from on-board devices such as GPS. Commodity GPS devices can provide meter-level positioning accuracy and nanosecond-level timing accuracy [1]. Many VANET applications rely on the information embedded in beacons. For example, in the Cooperative Collision Warning (CCW) application, vehicles continuously monitor their neighbors' current position and speed to warn the driver of potential accidents. In this paper, we denote $B_t(x)$ as a beacon sent by vehicle x at time t. When there is no ambiguity in terms of the sender's identity or time, B_t or B is used for brevity. In contrast to the single-hop relevant applications, in *multi-hop relevant applications*, messages are sent on demand and to vehicles that are potentially multiple hops away. For example, when sensing congestion, the Congestion Road Notification (CRN) application will create a message indicating the location, time, and congestion level to warn distant vehicles of the jammed segment.

2.2 Cryptographic Primitives

In the remainder of this section, we provide background on One-Time Signatures (OTS) and an OTS scheme called Merkle tree signatures. The core ideas of applying OTS to VANET authentication are detailed in Section 4.

One-Time Signature (OTS). A One-Time Signature [21,27,30] is a special type of public key cryptosystem whose signature generation and verification are thousands of times faster than other signature schemes like ECDSA. However, OTS suffer from longer keys and signatures (thus higher communication overhead) and their signing key can only be used once to sign a single message.

For example, to sign a 1-bit message m, the OTS signer generates two key pairs: $\{pk_0, sk_0\}$ and $\{pk_1, sk_1\}$, where the private keys sk_0 and sk_1 are chosen at random, and the public keys are computed as $pk_0 = H(sk_0)$ and $pk_1 = H(sk_1)$, where $H(\cdot)$ is a one-way function without trapdoors, and thus it is computationally infeasible to recover sk_b from $H(sk_b)$. If the message to be signed is 0, the signer publishes its signature $S = sk_0$. If the message to be signed is



$h_0 = H(v_0||r_0)$
$h_1 = H(v_1||r_1)$
$h_2 = H(v_2||r_2)$
$h_3 = H(v_3||r_3)$
$h_{0-1} = H(h_0||h_1)$
$h_{2-3} = H(h_2||h_3)$
$PK = H(h_{0-1}||h_{2-3})$

Figure 1: Example of a Merkle signature tree. The signature of $m = v_1$ is $\{h_{2-3}, h_0, r_1\}$.

1, then $S = sk_1$. Hence when receiving a signature S on a 1-bit message m, the receiver can verify whether $H(S) = pk_m$. In practice, we can use cryptographic hash functions, such as SHA-1, to implement such one-way functions. For convenience, in the rest of this paper, we use the term *hash* to represent the process as well as the output of such a one-way function. Signing an x-bit message can be accomplished by signing each of these x bits separately, resulting in an overhead linear to the size of the message.

Merkle tree signatures. An alternative to processing each bit separately is to create a common public key over all possible message values using a Merkle hash tree, as shown in Fig. 1. The hash tree root is the public key PK and each leaf is the hash of the concatenation of one of the possible message values and a random value. Each inner vertex represents the hash of the concatenation of its children. After constructing this Merkle hash tree, the signer can obtain a signature as follows: the signature of a message v_x consists of the corresponding random value r_x and all *off-path* vertices from the leaf node to the root. Off-path vertices are the siblings to the ancestors of h_x. Merkle signatures provide non-repudiation because of the computational infeasibility of finding another message with the same off-path values. Fig. 1 illustrates the Merkle hash tree construction representing an OTS to sign a 2-bit message, which has four possible values: $v_0 = 00$, $v_1 = 01$, $v_2 = 10$, and $v_3 = 11$. r_0, r_1, r_2, r_3 are random values, and the arrows indicate inputs to the hash function H. Hence the signature of $m = v_2$ is $S = \{h_{0-1}, h_3, r_2\}$. To validate this signature, the receiver recomputes the root of the hash tree from the received signature S and the message m. Then the receiver verifies whether the root equals PK, announced previously by the signer. In this paper, we use this Merkle signature scheme as a cryptographic primitive to construct one of our flooding-resilient signature schemes.

Tradeoffs in broadcast authentication. Table 1 shows the overhead of OTS and ECDSA signatures, which represent two extreme designs in broadcast authentication: OTS enables fast verification with expanded signatures, whereas ECDSA signatures are short but require long verification time. Hence, the challenge is to enable *fast authentication with short signatures*.

3. PROBLEM DEFINITION

Our goal is to design signature schemes with fast verification to mitigate signature flooding in the context of VANETs. Signature flooding describes a scenario where a large number of signature verification requests overwhelm the receiver's computational resources. We consider signature-based broadcast authentication schemes that guarantee message authenticity and non-repudiation.

Threat Model. We consider signature flooding caused by "flash crowds" [16] (i.e., a large number of benign vehicles broadcasting valid signatures within the victim's radio range) and by one or more colluding attackers sending *invalid signatures*. We do not consider attackers flooding other vehicles with a high volume of valid signatures since the receivers can quickly identify the attackers and then blacklist them. For a given authentication scheme, the attackers may also trigger unnecessary message verification by performing scheme-specific actions. A vehicle under signature flooding may be unable to verify every received signature before the message's deadline due to limited computational resources. Jamming attacks can also prevent VANET participants from communicating but can be addressed by jamming defenses [7,23].

Per-signature overhead	ECDSA-256	OTS
Generation	7 ms	320μs
Verification	22 ms	160μs
Public key size	256 bits	320 hashes (1 hash)
Signature size	512 bits	160 hashes (160 hashes)

Table 1: Signature generation and verification time reported in VAST [37], assuming signing the SHA-1 hash of a message. Numbers in parentheses represent the overheads of the Merkle signature scheme, where the size of one hash is 160 bits in case of the SHA-1 hash function.

Desired properties. VANET broadcast authentication has two desired properties: *non-repudiation* and *timely verification*. With the non-repudiation property, the recipient can prove to a third party who is accountable for creating a message. Non-repudiation also implies authentication, such that the recipient can identify the sender and detect malicious injection or manipulation of packets. Timely signature verification is required as well because each VANET message has an expiration time (or deadline) by which it should be verified. Single-hop applications often have a shorter deadline than multi-hop applications.

4. FAST AUTHENTICATION (FastAuth)

This section presents FastAuth, an efficient One-Time Signature (OTS) scheme to authenticate beacon messages. The novelty in FastAuth is the design of a *Chained Huffman Hash Tree (CHT)*, which leverages the predictability in vehicle mobility to generate small signatures.

As introduced in Section 2.1, a VANET-enabled vehicle broadcasts beacon messages at a rate of 10Hz to inform nearby vehicles of its position and velocity, enabling safety applications to alert drivers of potential accidents. Every beacon has to be verified upon its reception because the beacon may contain decisive and urgent safety information. However, the current VANET signature standard [15] (i.e., signing every beacon using ECDSA) is computationally expensive, whereas conventional OTS enables instant verification with increased communication overhead. Motivated by this unique challenge, the goal of FastAuth is to achieve fast authentication with short signatures.

4.1 Insights to Generate Short OTS Signatures

The intuition on how predictability in vehicle movements helps shorten OTS signatures is as follows. We observe that the *entropy* of a beacon is relatively low from the sender's point of view. In other words, the sender can predict what will be sent in subsequent beacons with high probability. For example, the timestamp can be pre-determined given that beacons are sent regularly (every 0.1 second), and the velocity is largely deterministic given the trajectory.

However, predicting the trajectory is non-trivial. It is difficult to determine a definite future position due to a vehicle's degrees of freedom. Fortunately, a prediction is possible because of the following reasons: 1) The laws of physics and road topology restrict the possible future locations. For example, a vehicle traveling slower than 80 miles per hour can move at most 3.3 meters in 0.1s. 2) Most of the time the vehicle is likely to move forward along the road rather than backward or sideway. 3) Due to the inherent positioning inaccuracy introduced by positioning devices (e.g., 2 meters inaccuracy in commodity GPS devices), we can round a location to a coarser-grained numerical representation to further reduce the number of candidate locations. For example, rounding location to meters only inflates the positioning inaccuracy from 2 meters to 2.5 meters, which is still acceptable for safety applications. Consequently, a small number of positions will occur with high probability and other positions are unlikely. Taking such a probability distribution into account, we are able to construct a one-time signature scheme using coding theory to shorten the signatures. Particularly, FastAuth adopts Huffman coding [14], an encoding algorithm minimizing the expected length of encoded messages. That is, FastAuth explores a new trade-off point in the design space of broadcast authentication: the size of a FastAuth signature depends on how well the signer can predict its own message content.

4.2 Protocol Overview

At a high level, each vehicle in FastAuth divides its timeline into a sequence of *prediction intervals*. In each prediction interval, a vehicle performs three steps: *Beacon Prediction, Key Pair Construction*, and *Signature Broadcast*. We provide an overview of these steps as follows.

Beacon Prediction (Sec. 4.3) At the beginning of a prediction interval, each vehicle predicts its beacon messages for the next I beacons. To do so, vehicles model the probability distribution of the distance vector between two consecutive beacons based on information of the past trajectory. For example, in Fig. 2(a), the vehicle predicts that it will move forward (D_f) with probability 0.5.

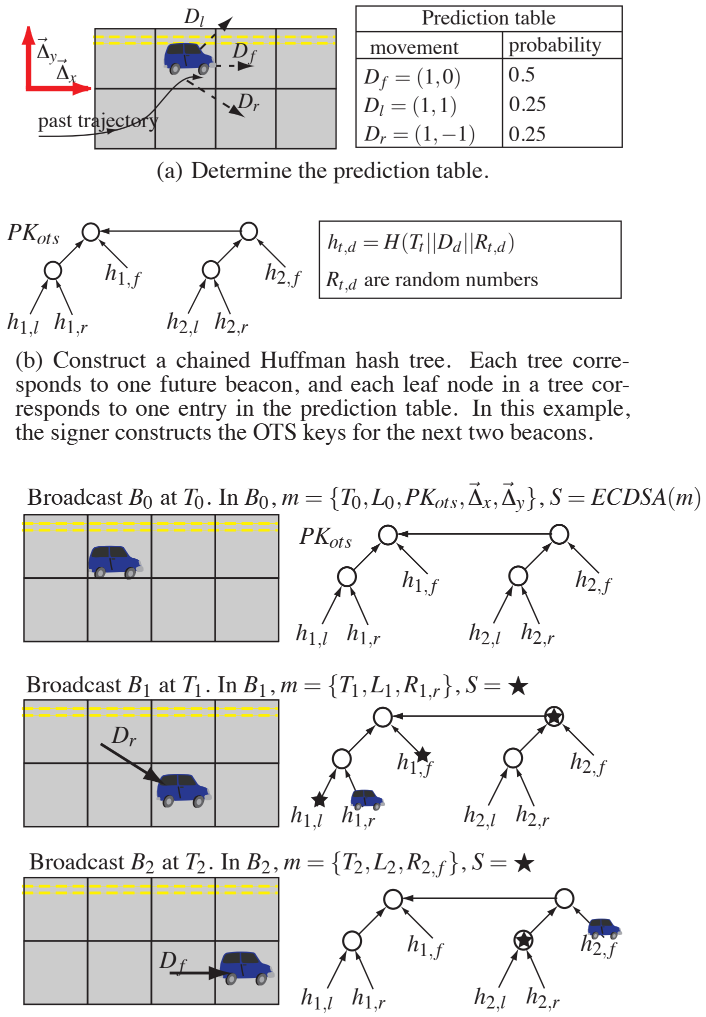
Key Pair Construction (Sec. 4.4) Before sending any beacon message in this interval, the vehicle needs to construct an OTS public key and one interval worth of OTS private keys. We propose a *chained Huffman hash tree (CHT)*, which ties these pre-computed keys together in a fashion that minimizes the size of signatures and generates a single public key, as shown in Fig. 2(b).

Signature Broadcast (Sec. 4.5) After beacon prediction and key construction, a vehicle signs its OTS public key using ECDSA signatures and then broadcasts this ECDSA signed public key PK_{ots} along with the first beacon in this prediction interval. When a vehicle moves to a position L_i at time T_i, it sends out a beacon B_i with this time and position information. Moreover, to maintain a high beacon update frequency during severe packet loss, we integrate Forward Error Correction [25, 33] into FastAuth to recover lost beacons.

4.3 Beacon Prediction

Since position is the main source of uncertainty in beacon message contents, we discuss how the beacon sender can predict and encode its own future positions. Specifically, FastAuth requires the sender to model the probability distribution of the *movement* (or the distance vector) between every two consecutive beacons, B_{i-1} and B_i. The output of this step is a prediction table G_i in which each entry represents one possible movement between time T_{i-1} and T_i and the probability of making this movement.

Representing positions using a local coordinate. To compress the amount of information, the sender expresses its future positions using a coarse-grained local coordinate. The origin of this local coordinate is placed at the beginning position of the current prediction interval (i.e., L_0). Then every future position L_i in this interval can be approximated by



D_l
$\vec{\Delta}_y$
$\vec{\Delta}_x$
D_f
D_r
past trajectory

Prediction table	
movement	probability
$D_f = (1,0)$	0.5
$D_l = (1,1)$	0.25
$D_r = (1,-1)$	0.25

(a) Determine the prediction table.

PK_{ots}
$h_{1,f}$
$h_{2,f}$
$h_{1,l}$ $h_{1,r}$
$h_{2,l}$ $h_{2,r}$
$h_{t,d} = H(T_t||D_d||R_{t,d})$
$R_{t,d}$ are random numbers

(b) Construct a chained Huffman hash tree. Each tree corresponds to one future beacon, and each leaf node in a tree corresponds to one entry in the prediction table. In this example, the signer constructs the OTS keys for the next two beacons.

Broadcast B_0 at T_0. In B_0, $m = \{T_0, L_0, PK_{ots}, \vec{\Delta}_x, \vec{\Delta}_y\}$, $S = ECDSA(m)$
PK_{ots}
$h_{1,f}$
$h_{2,f}$
$h_{1,l}$ $h_{1,r}$
$h_{2,l}$ $h_{2,r}$
Broadcast B_1 at T_1. In B_1, $m = \{T_1, L_1, R_{1,r}\}$, $S = \bigstar$
D_r
$h_{1,f}$
$h_{2,f}$
$h_{1,l}$ $h_{1,r}$
$h_{2,l}$ $h_{2,r}$
Broadcast B_2 at T_2. In B_2, $m = \{T_2, L_2, R_{2,f}\}$, $S = \bigstar$
D_f
$h_{1,f}$
$h_{2,f}$
$h_{1,l}$ $h_{1,r}$
$h_{2,l}$ $h_{2,r}$

(c) Beacon broadcast. To verify B_2, the receiver reconstructs the tree root for T_1 using the information in B_1 and B_2, and checks if the root matches the one signed in B_1.

Figure 2: Example of FastAuth.

$$L_i \approx L_0 + x_i\vec{\Delta}_x + y_i\vec{\Delta}_y, \text{where}$$

- $\vec{\Delta}_x$ is a vector parallel with the instant velocity at T_0,
- $\vec{\Delta}_y$ is perpendicular to $\vec{\Delta}_x$,
- x_i and y_i are rounded to integers,
- and the scalar of each vector (i.e., $|\vec{\Delta}_x|$ and $|\vec{\Delta}_y|$) is chosen by the vehicle to achieve a desired level of positioning accuracy.

For example, $|\vec{\Delta}_x|$ and $|\vec{\Delta}_y|$ can be set to 2 meters, which is the accuracy of commodity GPS devices, because a finer representation would contain more noise than useful information. Hence, each movement from time T_{i-1} to T_i

$$D_i = L_i - L_{i-1} = (x_i - x_{i-1})\vec{\Delta}_x + (y_i - y_{i-1})\vec{\Delta}_y$$

can be encoded as a pair of integers $(x_i - x_{i-1}, y_i - y_{i-1})$.

Constructing prediction tables. A prediction table G_i maps a movement D_i in the new coordinate to a probability of making such a movement. For example in Fig. 2(a), an entry $[(1,1), 0.25]$ in the prediction table means that the probability of moving one unit along

$\vec{\Delta}_x$ and one unit along $\vec{\Delta}_y$ is 0.25. A vehicle may have different prediction tables from other vehicles. The prediction tables can also vary with time, location, and speed. However, accurately modeling a vehicle's mobility is outside the scope of this paper; our focus is to design a broadcast signature scheme that performs better as the prediction accuracy increases. For illustration purposes, we discuss two possible approaches to generate a prediction table in Section 6.

Efficient Probabilistic modeling of future positions. A prediction table models a vehicle's movement in one beacon interval. By combining several prediction tables, we are able to model the vehicle's position not only in the next beacon but in many succeeding beacons. Let a prediction table G_i be a random variable representing the distance vector from L_{i-1} to L_i, and thus $\sum_{i=1}^{i=I} G_i$ is the random variable representing the movement from L_0 to L_I.

To see the advantage of encoding movements in a local coordinate, consider a simple scenario where a vehicle either moves forward 1 unit or stays at the same location during each beacon interval. In FastAuth, only 1-bit information (moving or staying) needs to be announced in each beacon. However, without considering the dependency between beacons, the vehicle has to encode i possible positions for beacon B_i at $T = i$, because the vehicle can be anywhere between position 0 and i.

4.4 OTS Key Construction

Before sending any beacons, a vehicle needs to generate OTS private keys for signing and the public verification key. Given the estimated future locations obtained in the previous step, our proposed *Chained Huffman Hash Trees* (CHT) structure minimizes the signature size by leveraging Huffman coding.

Combining Merkle and Huffman trees. A Merkle hash tree is a binary tree in which each leaf is assigned a value and the value for an inner node is the hash of its children. As discussed in Sec. 2.2, we can construct an OTS scheme using Merkle trees. Similarly, a Huffman tree is a binary tree data structure. In contrast to having values at leaves, each leaf node in a Huffman tree is associated with a probability. The sum of all leaves' probabilities adds up to one. For any pair of leaves in a Huffman tree, the leaf with a higher probability is never located deeper than the other leaf associated with a lower probability. Huffman trees provide the property of minimal expected depth of leaves. Hence, we can integrate these two trees into one *Huffman hash tree*, in which each leaf is assigned a probability p and a value h, since Huffman coding determines the organization of a tree and Merkle hash construction determines the value of tree vertices. The tree organization follows the rules of building Huffman trees and the value of each inner node is determined as in Merkle trees. In the VANET context, for each entry $[D_d, p]$ in the prediction table G_t (which means that from T_{t-1} to T_t, the vehicle will move to location $L_{t-1} + D_d$ with probability p), there is a leaf in the Huffman hash tree labeled with $(p, H(T_t||D_d||R_{t,d}))$, where $R_{t,d}$ is a random value to prevent signature forgery.

Chaining Huffman hash trees. A CHT contains I Huffman hash trees linked together to further reduce the computational overhead in verifying signed beacons. The roots of these trees are chained together chronologically and the anchor of the chain is the public key PK_{ots} for the current prediction interval. Fig. 2(b) illustrates the resulting chained tree structure. While the root of a single hash tree is computed by hashing the concatenation of its two children lc_t and rc_t ($root_t = H(lc_t||rc_t)$), in the chained structure

$$root_t = H(lc_t||rc_t||root_{t+1}) \tag{2}$$

for all $1 \le t \le I-1$. Then the public key PK_{ots}, or the hash anchor, is $PK_{ots} = root_1$.

4.5 Signature Broadcast

After constructing the prediction table using a local coordinate and generating the OTS public key PK_{ots}, the vehicle broadcasts its first beacon containing its public key and the parameters of its local coordinate system, i.e., the first beacon in a prediction interval is $B_0 = \{m_0, S(m_0), cert\}$, where $m_0 = \{T_0, L_0, PK_{ots}, \vec{\Delta}_x, \vec{\Delta}_y\}$ is signed using ECDSA.

To sign a beacon for position L_i and time T_i, the vehicle locates the leaf node corresponding to L_i and T_i, and broadcasts the *off-path* hash values of this leaf as the signature. Off-path nodes are defined as the siblings of nodes on the path from this leaf to $root_i$. Fig. 2(c) illustrates which hashes should be selected as the signature at each time. The car indicates the leaf associated with the current time and location. The nodes labeled with "★" are the off-path nodes that "sign" the current message. In this example, at T_1 the signer has moved to $L_1 = L_0 + D_r$, associated with $h_{1,r}$. Hence, B_1's signature includes the off-path hashes: $h_{1,l}$, $h_{1,f}$, and the root of the next hash tree, $root_2$. To verify the signature, a receiver of B_1 reconstructs $h_{1,r}$ from the message and computes the current hash tree root, $root_i$, from the off-path hashes. If the root matches PK_{ots}, the receiver is convinced that the sender moves D_r distance from T_0 to T_1 thus being located at position $L_0 + D_r$. Similarly, B_2's signature is the sibling node of $h_{2,f}$. Generally to verify signed B_i, the receiver reconstructs the current tree root, $root_i$, and checks whether Equation (2) holds given the hash values in the previous hash tree.

Dealing with packet loss. One drawback of the CHT structure is that once the receiver misses a beacon, it cannot verify any subsequent beacons in the current prediction interval, because the verification of the current beacon relies on the root of the previous beacon. Hence on average missing one beacon prevents verification of $I/2$ beacons. In contrast, when using ECDSA missing one beacon has no impact on verifying the future beacons. Unfortunately, packet loss is common in wireless networks. Especially in VANETs, the loss rate can be up to 30% in a benign environment, and up to 60% in a congested environment [3]. Hence it is important to deal with packet loss to reduce the number of unverifiable beacons. To mitigate the impact of packet loss, FastAuth adds redundancy into beacons using Reed-Solomon coding [33] such that when a beacon B_i is missing, the reception of u out of the succeeding w beacons (i.e., B_{i+1}, B_{i+2} ..., B_{i+w}) can help reconstruct B_i. Let $RS(w,u)$ be the Reed-Solomon coding process, e.g., encoding B_i using $RS(3,2)$ generates three codeblocks, r_1, r_2, r_3, and any two of them together can reconstruct B_i. The size of a codeblock is $|B_i|/2$. To enable error correction, the vehicle broadcasts r_1 along with B_{i+1}, r_2 with B_{i+2}, and so on. The higher the loss rate is, the more redundancy should be added into beacons, which can be achieved by increasing w/u at the cost of higher communication overhead. Hence, vehicles can adjust the RS code parameters to achieve a desired level of performance given an estimated packet loss rate.

Public key rebinding. PK_{ots} is only sent at the beginning of a prediction interval. If a vehicle X encounters vehicle Y after Y broadcasts its current OTS public key, X will be unable to verify Y's beacon until the next interval, which may take several seconds and X could already physically encounter Y. To overcome this drawback, every I_E beacons vehicle Y signs its beacon by ECDSA in addition to OTS. Hence, the receiver can initiate authentication from this additional trust anchor. Moreover, this mechanism can bound the beacon update frequency in face of high loss rate.

Dynamic interval. If a vehicle moves outside the area covered by its prediction table, the corresponding OTS will not exist in the CHT. For example, in Fig. 2(c), if the vehicle remains stationary between T_1 and T_2, no OTS signature exists for B_2 due to the absence of the corresponding entry (i.e., $(0,0)$) in the prediction table. Hence, instead of having a fixed interval, we allow vehicles to evoke a new predication process whenever the current position deviates outside the coverage of the prediction table. Terminating the current prediction interval and restarting a new one greatly increases the computational overhead. Fortunately, early termination of an interval is unlikely to occur when the prediction table covers most of the possible displacements in one beacon interval, as designed in Fig. 4.

4.6 Preliminary Analysis

The bandwidth saving in FastAuth depends on the accuracy of the location prediction. Hence in the rest of this section we discuss and quantify how the position prediction accuracy affects FastAuth's signature size. In section 6 we perform an evaluation using real traffic traces.

We denote by Φ the trajectory prediction function used by a vehicle, and by $\mathscr{L}_i$ the actual position of a vehicle at time T_i. $\Phi(T_i, x)$ returns the probability that the vehicle will be at position x at time T_i. On one extreme, when Φ is a prefect prediction function (i.e., $\Phi(T_i, \mathscr{L}_i) = 1$) the vehicle consumes the minimum possible bandwidth because for each beacon the vehicle only needs to send one hash to acknowledge that it is still moving along the estimated trajectory. On the other extreme, when Φ outputs a completely wrong prediction (i.e., $\Phi(T_i, \mathscr{L}_i) = 0$), the vehicle has to perform re-prediction and send an ECDSA signed beacon every time.

Now we consider a general case where $\Phi(T_i, \mathscr{L}_i) = p$ $(0 < p < 1)$ with n non-zero values in the prediction function's input domain (i.e., n possible movements). An accurate prediction function has a high p and a small n. To simplify the analysis, we assume except $\mathscr{L}_i$, every position is associated with the same probability $\frac{1-p}{n-1}$. Also let $S_\Phi(T_i)$ be the signature sent at T_i and $|S_\Phi(T_i)|$ be the size of the signature (in number of hashes). Hence (by Huffman coding), when $p > \frac{1-p}{n-1}$, $|S_\Phi(T_i)| \approxeq \log_2 n - \log_2(pn/(1-p)) \approxeq \log_2((1-p)/p)$. On the other hand, if $p \leq \frac{1-p}{n-1}$, $|S_\Phi(T_i)| \approxeq \log_2 n$. Taken together, we can conclude that an inaccurate prediction function (i.e., a low p or a large n) results in longer signatures.

5. Selective Authentication (SelAuth)

In Section 4, we presented FastAuth, a signature scheme supporting instant verification of single-hop messages (beacons) by exploiting the low entropy of beacons. In this section, we switch our focus to multi-hop VANET applications, in which messages often traverse multiple hops before expiration. In multi-hop applications, vehicles only need to check a signature when the message needs to be acted on (verify on demand). Hence for most messages vehicles can forward them without verification to preserve their limited computational resources. However, such a verify-on-demand approach is vulnerable to communication-based denial of service attacks where the attacker sends a large number of bogus signatures to exhaust the available bandwidth, because a single invalid signature is likely to be relayed and duplicated many times before being detected at the destination(s). On the other hand, checking every signature (Verify-All) blocks invalid signatures as close to the source as possible but is computationally expensive.

In response to the challenge for designing an efficient verification scheme for multi-hop applications, we present SelAuth, a signature verification protocol that can quickly block the spread of invalid signatures without checking all incoming signatures at every hop. In essence, we observe that when the receiver knows more about a received signature, i.e., high certainty (low entropy) on the signature's validity, the signature can be checked with a lower probability. In particular, SelAuth uses *neighbor identification* to avoid impersonation and *per-neighbor verification probability*, adjusted dynamically as wrong signatures are received, to achieve isolation. Neighbors of a vehicle can directly communicate with the vehicle. Let p_{xy} denote the probability that vehicle y checks a packet forwarded by vehicle x. x and y are neighbors. The goal is to update p_{xy} so that $p_{xy} \rightarrow 0$ for legitimate x, and $p_{xy} \rightarrow 1$ for malicious x. After a short period of time the network should converge to a state where neighbors of malicious vehicles check everything whereas others check nothing. SelAuth also uses *warning pushback* to accelerate the isolation of malicious vehicles.

As in many other probabilistic verification schemes [10,19,34,38], vehicles running SelAuth verify an incoming message with a certain probability in order to help detect invalid signatures. However, an important difference is that in prior work such a probability depends solely on the local status of the receiver, regardless where this message is from or whether other vehicles have verified this message. In contrast to having the vehicle at each hop making its authentication decision independently, SelAuth leverages auxiliary information shared between neighbors to facilitate the probability adjustment for fast and efficient isolation.

5.1 Protocol Overview

The core components of SelAuth are forwarder identification and warning pushback, enabling SelAuth to converge faster than other probabilistic verification schemes and be more resource-efficient than the Verify-All approach.

Forwarder identification. Forwarder (or neighbor) identification enables the receiver of a message to efficiently identify which of its neighbors sent or forwarded this message. Digital signatures provide this property but are computationally expensive. As a result, we achieve forwarder identification using symmetric cryptographic operations, as described in Section 5.2. Moreover, forwarder identification holds vehicles accountable for their forwarding behavior and enables a deployment of **per-forwarder verification probability**. For example, if A receives m_b forwarded by B and m_c forwarded by C, A will verify m_b with probability p_{ba} and m_c with probability p_{ca}. These probabilities get updated over time; eventually the links incident to malicious nodes have a high verification probability, and a bad node that has forwarded too many invalid signatures is punished by being blocked from communication.

Warning pushback. In SelAuth, vehicles detecting an invalid signature will initiate a *Complaint* message to warn vehicles at the previous hop. If a vehicle at the previous hop agrees, it will issue a Complaint as well. As a Complaint propagates back to the originator of the invalid signature, every vehicle that had forwarded this signature can be warned and then increases its verification probability accordingly. Hence, it is also called a *Pushback* message as it is pushed towards the originator.

Protocol flow. Fig. 3 shows SelAuth's protocol flow when y receives a message from a neighbor x. Let p_{xy} be vehicle y's verification probability on messages forwarded by x. WLOG, we assume y is legitimate and x may be either legitimate or malicious. p_{xy} is undefined when y is malicious because y can behave arbitrarily. Let $[M]_x$ denote a message forwarded by x. $[M, ID_R]_x$, where $M = \{m, S(m), cert\}$ is a standard VANET message and ID_R represents the one-hop receiver(s): ID_R is the target ID for pushback and $ID_R = *$ otherwise for broadcast. y first checks whether M has been received before (e.g., sent by another neighbor) and identifies the forwarder. If M is new and sent by a known neighbor, y checks the signature $S(m)$ with probability p_{xy}. If $S(m)$ is valid or the check is bypassed, y rebroadcasts the message and decreases p_{xy}; otherwise

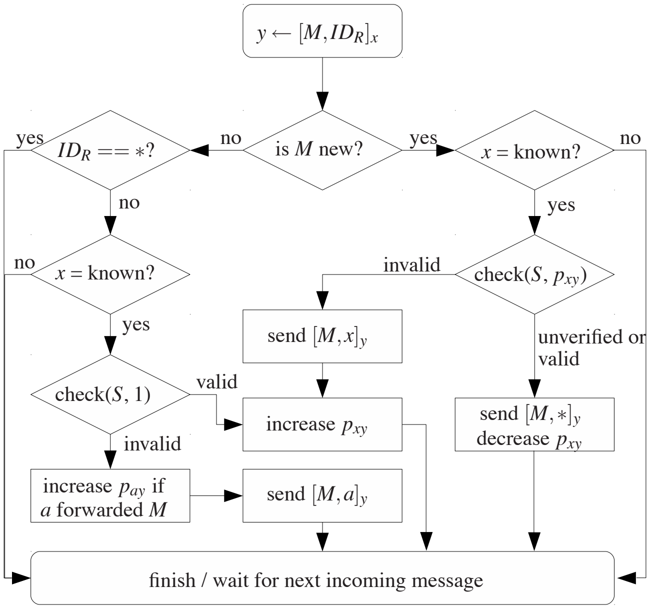


Figure 3: Flowchart of SelAuth. We discuss how probabilities are initialized and adjusted in Sec. 5.2.

p_{xy} increases. We discuss the details of probability initialization and adjustment in Sec. 5.2. If M is a pushback from a known forwarder complaining about an old message, y checks whether the complaint is true: if $S(m)$ is really invalid, y finds out who had forwarded M to it and increases the probability accordingly; if $S(m)$ is in fact valid, y increases p_{xy} to punish x for lying.

5.2 Realizing Forwarder Identification

Forwarder identification can be achieved using authentication. At first glance, this sounds like a circular problem: our SelAuth authentication scheme relies on forwarder identification, which requires authentication as well. However, the difference is that SelAuth is a signature mechanism for multi-hop applications, whereas forwarder identification requires only single-hop authentication without non-repudiation, because for forwarder identification, a vehicle only needs to prove its identity to its one-hop neighbors rather than a third party.

For such single-hop authentication, a digital signature is too expensive to generate and verify. Also, FastAuth is inapplicable to be applied directly because messages in multi-hop relevant applications are infrequent and hard to predict. Fortunately, we observe that the TESLA authentication scheme [31] can be applied here with few modifications. TESLA requires loose time synchronization, which is naturally supported in VANETs where messages are timestamped with nanosecond accuracy by GPS-equipped vehicles. TESLA constructs self-authenticating one-way hash chains, in which each value can be authenticated efficiently using the previous value in its hash chain. The anchor of a hash chain is digitally signed as a root of trust for bootstrapping. Specifically, in SelAuth, we leverage FastAuth to sign trust anchors, since these anchors can be pre-computed thus known to the sender before constructing beacons. For simplicity, in the following we assume the hash values are revealed in sequence in beacons, i.e., B_i contains a_{i-1} and $a_{i-1} = f(a_i)$, where f is a one-way function. One benefit of including each hash-chain value in beacons is that a vehicle can start the forwarder identification process even if the vehicle misses the initial trust anchor of the hash chain. In practice, hash-chain values can be sent independently from beacons. A vehicle uses a_i to derived a *cryptographic forwarder ID* for each packet it sends between B_{i-1} and B_i. A cryptographic ID for a message M sent between B_{i-1} and B_i is $\text{MAC}_{f'(a_i)}(M)$, where MAC is a Message Authentication Code and f' is a one-way function different from f. A cryptographic forwarder ID is appended to its associated message to make the message's forwarder identifiable. After receiving $\text{MAC}_{f'(a_{i-1})}(M)$ and B_i, the receiver can be convinced that M forwarded by the sender of and B_i. Using TESLA for forwarder identification, SelAuth introduces verification delay because a receiver has to wait for the disclosure of hash-chain values for verification. However, a small delay is acceptable in flooding-based multi-hop communication where timing is not critical.

5.3 Implementation Decisions

Memory-efficient message status checking. To identify previously forwarded messages, each vehicle maintains one Bloom Filter [4] for valid messages and one for invalid messages, and keeps per-neighbor Bloom Filters for unverified messages from each of the neighbors. Hence when getting a Complaint about a previously received but unverified message M, the vehicle can identify the forwarder of M and increases the probability accordingly. Each vehicle also keeps a Bloom Filter for Complaints to prevent replay attacks that would further increase the verification probability. Also, the filters are reset periodically to remove stale messages. A Bloom Filter is a space-efficient data structure enabling fast membership checking. If a message M has been added into a Bloom Filter BF, checking $BF(M)$ will always return true (match found); if a message M' has never been added into the Bloom Filter, checking $BF(M')$ will return false (no match) with very high probability. It is called a false positive when a Bloom Filter falsely returns a match. When a false positive occurs, vehicles may wrongly drop a new message. To prevent malicious vehicles from deliberately causing more false positives (e.g., by sending out a small number of properly crafted messages to fill the filter), every Bloom Filter should be kept private or keyed so that only the owner who picked the key k can evaluate the function $BF_k(M)$. Although false positives have a similar effect as packet loss, they have a limited impact on SelAuth because the false positive rate (e.g., 1%) is often much lower than the packet loss rate (e.g., 30%).

Probability initialization and adjustment. There are many ways to assign an initial value p_0 to p_{xy} and to adjust it after the initial assignment. For example, a vehicle can assign an identical initial probability to every newly identified neighbor. The adjustment function can be linear (the same amount of increment and decrement), MIAD (multiplicative increase and additional decrease), or a step function (jump to 1 and come back to p_0 when receiving a threshold number of valid signatures). We use a step function as suggested in the prior work [34] to accelerate the isolation process.

Probabilistic pushback. One invalid packet can trigger more than one pushback. However, only one pushback is needed to warn of the bad packet. To avoid unnecessary communication we implement *probabilistic pushback*, in which a vehicle y sends a pushback complaining a packet previously forwarded by x with a probability $1 - p_{xy}$, in contrast to initiating a pushback whenever detecting an invalid packet. The idea is that the lower the verification probability p_{xy} the more likely that the vehicle x is benign and needs the warning of the presence of malicious entities, whereas a high verification probability implies that x may be malicious and thus it will be a waste to send warnings to x.

Dealing with consistent attackers. To deal with consistent attackers who never send valid signatures, we always verify the first packet from a newly encountered neighbor. Moreover, if x's misbehavior persists, y should block the identified malicious forwarder by dropping all the traffic from x or invalidate x's certificate by reporting x to an authority.

Delayed verification. On one hand, we want to block invalid traffic from malicious senders. On the other hand, we also want to retain high throughput by taking advantage of these bad nodes, i.e., by permitting them to forward packets. To achieve this, SelAuth delays the verification of messages from malicious nodes. Delayed signature verification can help because a vehicle is likely to have received duplicated messages due to the broadcast nature. Delayed verification can address the case where a vehicle has only one neighbor but the neighbor is malicious. Rather than dropping all packets forwarded by the malicious neighbor and making itself get disconnected from the network, the vehicle may want to sacrifice some security for availability.

5.4 Preliminary Analysis

A full simulation of SelAuth can be found in Section 7. In this analysis, we consider a simple line model where vehicles are placed at location 0, l, $2l$, $\cdots$. The communication range is ld, so there are d next-hop vehicles that forward messages ahead. The initial verification probability is p for all vehicles. The attacker sends n invalid signatures during each time interval. Hence the probability that a receiver of the n invalid signatures does not catch any of them (and thus forward all of the messages) is $\alpha = (1-p)^n$. We investigate the case with an attacker at the origin. Note that SelAuth always checks the first message from a new neighbor, and thus the attacker's optimal strategy is to behave at the beginning ($t = 0$) by sending one valid signature and attacks after $t = 1$.

To quantify the impact level, let $I(t,h)$ be the expected number of invalid signatures forwarded at time t by vehicles h hops away from the attacker. Note that $I(t,h)$ is defined only when $0 \leq h \leq t$. Hence we can express $I(t,h)$ as:

$$I(t,h) = n(\alpha^{(t-h)h}\alpha^{\sum_{i=0}^{t-h}\lfloor\frac{t-h-i}{2}\rfloor}) = n\alpha^{1/4(t-h)(t+h+1)} \qquad (3)$$

where $\alpha^{(t-h)h}$ represents the probability that none of the vehicles in the first h hops check the attacker's packets sent in the first $t-h$ intervals. $\alpha^{\sum_{i=0}^{t-h}\lfloor\frac{t-h-i}{2}\rfloor}$ expresses the probability that the h-hop vehicles receive no pushback from other vehicles. This can be rephrased as "vehicles between $h+1$ and $\lfloor\frac{t-h-i+1}{2}\rfloor$ hops fail to check any bad packets sent during the first $\lfloor\frac{t-h-i+1}{2}\rfloor$ intervals". Similarly the impact level without the pushback mechanism is:

$$I_{FI}(t,h) = n\alpha^{(t-h)h}. \qquad (4)$$

The impact level in (3) decreases faster as t increases than in (4), as the pushback mechanism expedites the containment process.

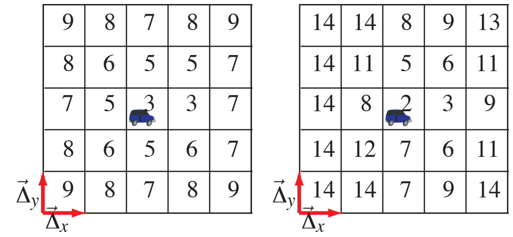
The analysis shows that SelAuth can converge promptly to a state where no invalid signatures will be relayed; an attacker at best can only cause local attacks congesting the neighbors' wireless channel within one-hop communication range, in contrast to large-scale DoS attacks that affect communication multiple hops away.

6. EVALUATION: Fast Authentication

To evaluate FastAuth, we consider a sender vehicle sending beacons and a receiver vehicle receiving these beacons with probability $1-\varepsilon$, where ε is the packet loss rate. The sender moves along real traffic traces collected by General Motors. The GM dataset includes four traces and each of them was generated by a vehicle driving along a 2-mile path for about 2 hours. Though the trajectory is pre-defined for the purpose of simulation, the sender can only leverage its past trajectory to predict future location. We simulate FastAuth and SelAuth (shown in Section 7) separately to enable a clearer analysis.

parameter	sample value
ECDSA generation time	7ms
ECDSA verification time	22ms
hash operation time	1μs
beacon size	378 bytes
hash size	20 bytes
RS code	$RS(5,2)$
prediction interval (I)	100 (10s)
ECDSA rebinding interval (I_E)	20 (2s)
packet loss rate (ε)	0.3

Table 2: Notation and sample values.



9	8	7	8	9
8	6	5	5	7
7	5	3	3	7
8	6	5	6	7
9	8	7	8	9

(a) default

14	14	8	9	13
14	11	5	6	11
14	8	2	3	9
14	12	7	6	11
14	14	7	9	14

(b) trained

Figure 4: Signature size (in number of hashes) when the vehicle moves to a particular block.

6.1 Evaluation Settings

Table 2 lists the parameters and sample values commonly used in VANET simulations [11, 37]. In addition to these parameters, FastAuth requires a prediction table to model future location. In practice, car manufacturers or VANET application providers can apply advanced physics models and well-analyzed traffic statistics to construct the prediction table. For the purpose of simulation, however, we consider two methods to construct the table. For both, we build a prediction table large enough to cover most movements made by a vehicle in one beacon interval (i.e., 0.1 second), given that the maximum speed limit in US is 80 miles/h or 129 km/h. The block unit is set to 2 meters given the positioning accuracy of commodity GPS systems. The first method considers the worst case where only a default prediction model is available, and allocates the probabilities using the principle of *"the nearer the distance, the larger the probability"*, as shown in Fig. 4(a). The other is to use the first half of each trace as the training data and evaluate FastAuth using the second half of trace. For each movement in the prediction table, its probability is determined by how often the vehicle made such a movement in the training data. This prediction table is shown in Fig. 4(b).

6.2 Simulation Results

In this simulation, we discuss the impact of the prediction interval (I), RS coding, packet loss rate (ε), and rebinding interval (I_E) on FastAuth. In particular, we present the performance ratio of FastAuth to ECDSA. Table 2 summarizes the parameters used in the evaluation. We evaluate FastAuth using the following metrics: 1) sender/receiver computational overhead, 2) communication overhead (which reflects the accuracy of our location prediction model), and 3) average beacon update frequency (defined as the number of beacons successfully verified per second).

Fig. 5 shows the effectiveness of using trained and default ta-

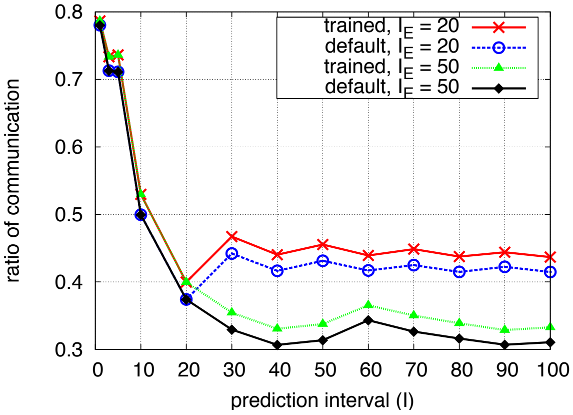


Figure 5: The effectiveness of trained and default tables as compared to ECDSA.

bles. Both models enable FastAuth to save more than a half of the bandwidth compared to ECDSA for $I \geq 10$. Interestingly, the default table performs better than using a short duration (one hour) of past trajectory as the indicator of future movement, since using the one-hour past trajectory without context is only a rough estimator. Hence, in the following we evaluate FastAuth using the default prediction table.

Fig. 6 shows the performance of FastAuth with various RS coding settings and rebinding intervals. Compared to ECDSA, signature generation in FastAuth (i.e., the sender's computation) is 20 times faster and verification (i.e., the receiver's computation) is 50 times faster. Also the communication ratio is only 20% – 40%. With a smaller rebinding interval ($I_E = 20$), the beacon update frequency is higher than using ECDSA because of the RS error correction coding. However, a small I_E increases communication and the sender's computational overhead. Also, more redundancy in the error correction code helps improve the update frequency at the cost of bandwidth consumption.

Fig. 7 shows the impact of the packet loss rate ε on the beacon update frequency and the sender's computational overhead; the other two metrics are constant with respect to ε. As ε increases, the ratio of the average beacon update frequency drops and the receiver's computational overhead increases. The result shows that FastAuth provides significant performance advantages even when ε is abnormally high, and the performance degrades gracefully as ε increases.

The result confirms that FastAuth can drastically reduce the computational overhead for both the sender and receiver. Specifically, our signature verification is 50 times faster and signature generation is 20 times faster than it using ECDSA. The communication overhead is reduced to 60% as well. Furthermore, FastAuth can achieve the same level of update frequency as ECDSA with the help of the error correction and key rebinding mechanisms.

7. EVALUATION: Selective Authentication

To evaluate SelAuth, we compare the performance of five signature verification schemes in multi-hop networks. We simulate these schemes using NS-2 [26] with the IEEE 802.11p MAC layer and Nakagami physical layer model [6], and synthesize vehicles' traces on a realistic road topology using SUMO [17]. We evaluate the performance of five signature verification schemes in multi-hop networks. These five schemes are summarized in the table below:

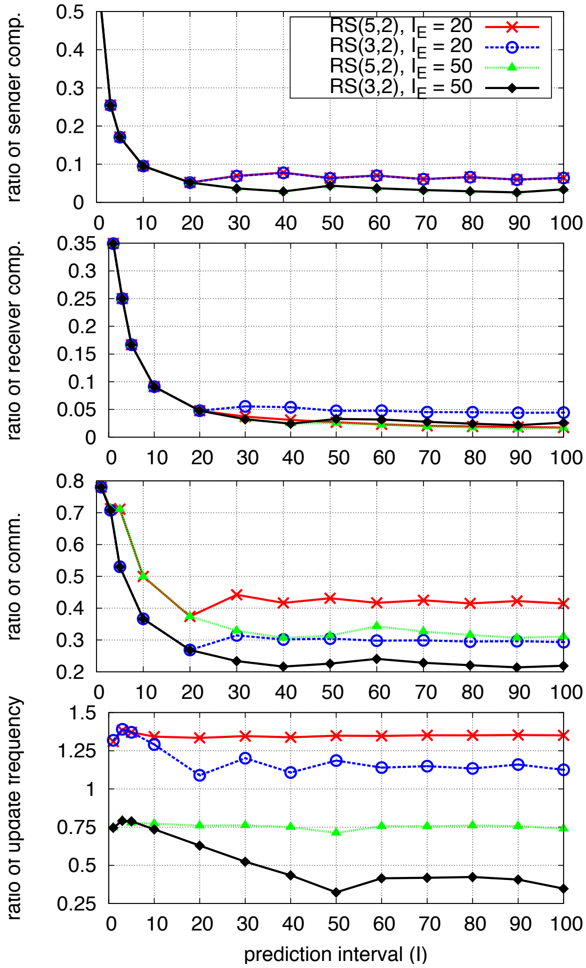


Figure 6: Evaluation based on real traffic traces.

scheme	probabilistic verification	per forwarder probability	probabilistic pushback
SelAuth	✓	✓	✓
Verify-All			
FI	✓	✓	
PB	✓		✓
AA [34,38]	✓		

Verify-All is a naive approach in which vehicles verify every incoming packet. *Forwarder identification only (FI)* uses a per-forwarder verification probability without pushback warnings, while *Pushback only (PB)* provides pushback warnings without a per-forwarder verification probability (i.e., a shared verification probability for every neighbor). In *Adaptive Message Authentication (AA)*, neither pushback nor forwarder identification is supported [34,38].

7.1 Evaluation Settings

We analyze SelAuth in two scenarios: a linear scenario validating the performance advantages and a real-world scenario demonstrating the practicability of deployment. In the linear scenario, 100 static vehicles are placed every 30 meters in a line and an attacker moves at a constant speed of 10m/s along the line. The simulation time is 50 seconds. The real-world scenario uses a road topology reconstructed from a 1km×1km downtown area of Manhattan. We simulate 336 vehicles with random traffic patterns generated by SUMO (average speed is 10m/s), and one attacker travels along a manually selected path to circle within the area. The simulation

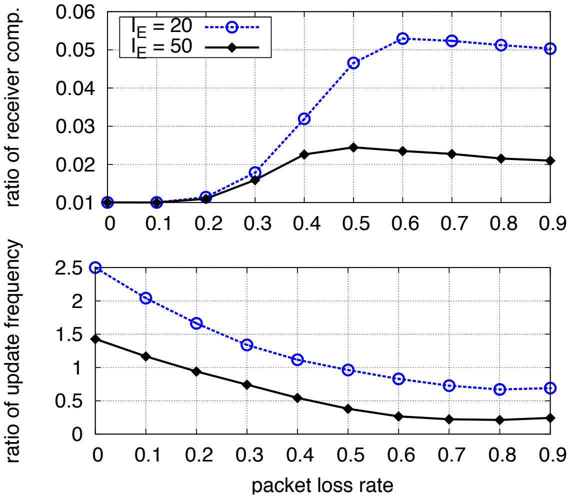


Figure 7: The impact of the packet loss rate.

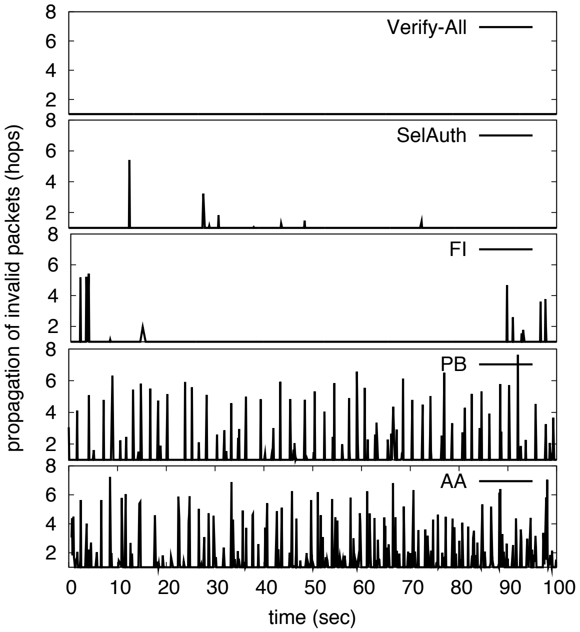


Figure 8: convergence speed.

time is 250 seconds, long enough for the attacker to traverse the entire area.

In both scenarios, every benign vehicle sends one 160-byte message/s for multi-hop applications and broadcasts 10 beacons/s to exchange information about forwarder identification. However, we do not consider the overhead caused by beacons in our comparison as the beacon overhead is nearly identical in all schemes. We evaluate schemes using three metrics: 1) *overall computational overhead*, expressed as the total number of ECDSA verification operations performed by vehicles during the time span of the simulation; 2) *overall communication overhead*, expressed by the total amount of packets sent by vehicles; 3) *average hop of invalid packets*, reflecting the convergence time.

7.2 Simulation Results

Fig. 8 evaluates the convergence speed and the effectiveness of isolation of each scheme in the linear scenario by inspecting how far the invalid packet sent at each time can propagate. In this setting the attacker sends packets at a rate of 10Hz and 80% of packets sent by the attacker are invalid. The simulation lasts for 100 seconds. Fig. 8 shows that in terms of the ability to block invalid packets, Verify-All > SelAuth > FI > PB > AA. Specifically SelAuth stops 99% of bad packets at the first hop. Although Verify-All can perfectly isolate all bad packets but it introduces an unacceptably high computational overhead as we will describe shortly.

We investigate the performance of SelAuth under different *initial probabilities* and *attack rates*. The initial probability affects overall computational overhead and the convergence speed. We also examine the effectiveness of the signature verification schemes under different sending rates of bad packets. Intuitively, a smart attacker would send valid packets occasionally to avoid detection and increase the chance of being removed from others' blacklist. The reason is that an attacker that has been detected is unable to resurgence by sending fake packets only. An easy and effective attack is to send fake packets probabilistically to avoid being blocked permanently by other nodes.

We study how different attack rates affect the schemes and the result is shown in the first row of Fig. 9. The initial verification probability is 0.3. The attacker sends 10 packets per second, and x% of the packets are invalid (x% is the attack rate). Given a constant packet-sending rate (both valid and invalid packets), an attacker with a higher fake packet sending rate can inject more fake packets but will be blocked by its neighbors for a longer time. In all schemes, the higher the attack rate, the higher the communication overhead. However, the average hop of invalid packets decreases. It confirms the intuition that it is more difficult to launch a stealthy attack when the attack rate is high.

The second row of Fig. 9 shows the relationship between various metrics and the initial probabilities for the real-world scenario (results for the linear scenario is similar so we omit them due to the space limitation). For the real-world scenario, we simulate different schemes in which the attacker's strategy "*performs best*" based on experiments in the linear case. Each data point on the figures represents the average of 20 runs. The attacker sends one packet every one second, and 8 out of 10 packets (selected randomly) are invalid. Generally, a smaller initial probability leads to lower computational overhead but slows down the reaction to invalid packets. The result shows that SelAuth causes the least computational overhead and provides close to optimal containment of invalid packets.

In sum, SelAuth has the lowest computational overhead among all schemes; it consumes only 15% of computational resources compared to the Verify-All approach. Also SelAuth achieves 99% of isolation of invalid packets, much higher compared to other probabilistic verification schemes.

8. RELATED WORK

The most closely related works [15, 34] are compared in Sections 6 and 7. This section gives a more comprehensive overview of works in broadcast authentication.

Researchers have explored the design space of broadcast authentication for years in various network settings, from resourceful environments like the Internet [5,9,10,18,19,29], to resource-limited and unreliable environments such as sensor networks [24, 31, 38] and vehicular networks [32,34,37]. Despite the diversity in broadcast authentication research, the common idea is to minimize the use of expensive cryptographic operations such as ECDSA digital signatures. There are two types of approaches: one is to move away from asymmetric cryptography to symmetric cryptography, and the

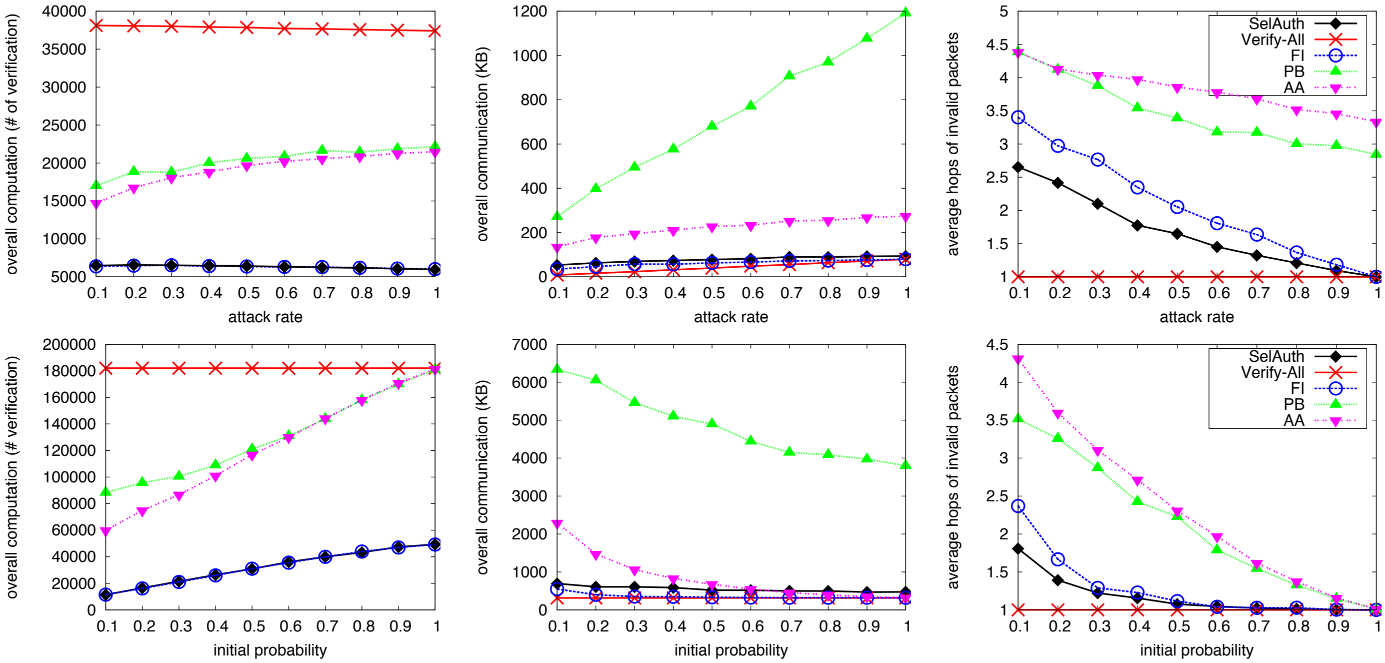


Figure 9: Upper row: the linear scenario with different attack rates. Lower row: the real-world scenario with different initial probabilities.

other is to reduce the amount of cryptographic work when there is no (or few) malicious activities in the network.

One-time signatures and one-way hash chains are two lightweight cryptographic primitives often used in constructing broadcast authentication schemes. As we discuss in the earlier section, one-time signatures generally provide fast authentication at the cost of short signatures [9,21,27,30]. Though FastAuth is also based on one-time signatures, it reduces the size of signatures by leveraging the predictability of a vehicle's trajectory. TESLA-based broadcast authentication schemes build one-way hash chains to authenticate symmetric keys [13,31,37]. However, TESLA-based schemes require a timestamping server to achieve the non-repudiation property. In TESLA, in order to verify packets, the receiver caches all incoming packets until the sender releases the associated MAC key that is used to sign these packets. Later on, the receiver can verify the cached packets using the released key. Such caching and key release processes prevent TESLA-based approaches from providing instant verification. Delayed verification is unfavorable in safety-related applications where the recipient would like to verify time-sensitive messages immediately. Moreover, even in multi-hop applications where delayed verification is acceptable, TESLA-based schemes cannot be applied directly since it is difficult to efficiently collect a signed trust anchor, which is used to bootstrap TESLA, from every vehicle.

The other type of efficient authentication schemes generally design mechanisms with adjustable overhead. Specifically, many works propose to use selective verification, where only a subset of messages is validated right upon reception and the rest is assumed valid until the actual verification is performed [10,19,22,34,38]. Again, selective verification performs well in some scenarios (e.g., multi-hop relevant VANET applications) but is inapplicable for others such as safety applications. For example, prior work has suggested that vehicles suppress less important beacons [36]. However, sending beacons only when emergency events occur is insufficient and dangerous because the absence of a beacon could mean either no abnormal events or that some attacker has successfully suppressed the beacons. Also, most of the prior work has difficulty in confining the spread of invalid messages originated by a mobile attacker.

The idea of integrating a Huffman coding tree with a Merkle hash tree has been discussed in the context of certificate revocation checking, where such an integration can reduce the size of response messages from an untrusted certificate repository [28]. However, since a Huffman hash tree is hard to adjust once being constructed, the checking scheme cannot handle repository update efficiently. In contrast, FastAuth, as a one-time signature scheme, inherently requires a new hash tree for every beacon and thus is free from such a limitation Also, invalid signature notification [22], an idea similar to the pushback warnings in SelAuth, has been proposed to help identify invalid but unverified messages. However, without teaming with SelAuth's forwarder identification mechanism, the attacker can abuse such notifications to cause denial of service.

9. CONCLUSION

We design and evaluate two flooding-resilient signature schemes for VANETs based on the observation that the overheads of broadcast authentication should match the entropy of messages. Our schemes are superior to previous work in their ability to provide efficient signature verification thus enabling VANET to operate in the presence of signature flooding. Specifically, FastAuth secures VANET periodic single-hop beacons by leveraging the beacon predictability. On the other hand, SelAuth secures multi-hop applications by prompt attack isolation. We believe that these two broadcast authentication schemes can mitigate signature flooding in many VANET applications. In any case, the techniques we propose in this paper bring VANETs one step closer to reality, as they enable resource-constrained devices to verify safety messages, which the current standard cannot achieve even in benign settings.

10. REFERENCES

[1] LEA-6 u-blox 6 GPS modules data sheet, 2010.

[2] BAI, F., KRISHNAN, H., SADEKAR, V., HOLLAND, G., AND ELBATT, T. Towards characterizing and classifying communication-based automotive applications from a wireless networking perspective. In *Proceedings of IEEE AutoNet* (2006).

[3] BAI, F., STANCIL, D. D., AND KRISHNAN, H. Toward understanding characteristics of dedicated short range communications from a perspective of vehicular network engineers. In *Proceedings of ACM MobiCom* (2010).

[4] BLOOM, B. H. Space/time trade-offs in hash coding with allowable errors. *Commun. ACM 13*, 7 (1970), 422–426.

[5] CANETTI, R., GARAY, J., ITKIS, G., MICCIANCIO, D., NAOR, M., AND PINKAS, B. Multicast security: A taxonomy and some efficient constructions. In *Proceedings of IEEE INFOCOMM* (1999).

[6] CHEN, Q., SCHMIDT-EISENLOHR, F., JIANG, D., TORRENT-MORENO, M., DELGROSSI, L., AND HARTENSTEIN, H. Overhaul of IEEE 802.11 modeling and simulation in ns-2. In *Proceedings of ACM MSWiM* (2007).

[7] CHIANG, J. T., AND HU, Y.-C. Cross-layer jamming detection and mitigation in wireless broadcast networks. In *Proceedings of ACM MobiCom* (2007).

[8] FRANCILLON, A., DANEV, B., AND CAPKUN, S. Relay attacks on passive keyless entry and start systems in modern cars. In *Proceedings of NDSS* (2010).

[9] GENNARO, R., AND ROHATGI, P. How to sign digital streams. In *Proceedings of CRYPTO* (1997).

[10] GUNTER, C. A., KHANNA, S., TAN, K., AND VENKATESH, S. DoS protection for reliably authenticated broadcast. In *Proceedings of NDSS* (2004).

[11] HAAS, J. J., AND HU, Y.-C. Communication requirements for crash avoidance. In *Proceedings of VANET* (2010).

[12] HSIAO, H.-C., STUDER, A., DUBEY, R., SHI, E., AND PERRIG, A. Efficient and secure threshold-based event validation for vanets. In *Proceedings of ACM Conference on Wireless Network Security (WiSec)* (2011).

[13] HU, Y.-C., AND LABERTEAUX, K. P. Strong VANET security on a budget. In *Proceedings of ESCAR* (2006).

[14] HUFFMAN, D. A. A method for the construction of minimum-redundancy codes. *Institute of Radio Engineers 40*, 9 (September 1952), 1098–1101.

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

[16] JUNG, J., KRISHNAMURTHY, B., AND RABINOVICH, M. Flash crowds and denial of service attacks: characterization and implications for CDNs and web sites. In *Proceedings of ACM WWW* (2002).

[17] K. DANIEL, H., GEORG, R., AND PETER, W. SUMO (Simulation of Urban MObility) - an open-source traffic simulation. In *Proceedings of MESM* (2002).

[18] KARLOF, C., SASTRY, N., LI, Y., PERRIG, A., AND TYGAR, J. D. Distillation codes and applications to DoS resistant multicast authentication. In *Proceedings of NDSS* (2004).

[19] KHANNA, S., VENKATESH, S. S., FATEMIEH, O., KHAN, F., AND GUNTER, C. A. Adaptive selective verification. In *Proceedings of IEEE INFOCOM* (2008).

[20] KOSCHER, K., CZESKIS, A., ROESNER, F., PATEL, S., KOHNO, T., CHECKOWAY, S., MCCOY, D., KANTOR, B., ANDERSON, D., SHACHAM, H., AND SAVAGE, S. Experimental security analysis of a modern automobile. In *Proceedings of IEEE Symposium on Security and Privacy* (2010).

[21] LAMPORT, L. Constructing digital signatures from a one-way function. Tech. rep., October 1979.

[22] LI, Z., AND CHIGAN, C. On resource-aware message verification in VANETs. In *Proceedings of IEEE ICC* (2010).

[23] LIU, Y., NING, P., DAI, H., AND LIU, A. Randomized differential DSSS: jamming-resistant wireless broadcast communication. In *Proceedings of IEEE INFOCOM* (2010).

[24] LUK, M., PERRIG, A., AND WHILLOCK, B. Seven cardinal properties of sensor network broadcast authentication. In *Proceedings of ACM workshop on Security of ad hoc and sensor networks (SASN)* (2006).

[25] MCAULEY, A. J. Reliable broadband communication using a burst erasure correcting code. *SIGCOMM Comput. Commun. Rev. 20*, 4 (1990), 297–306.

[26] MCCANNE, S., FLOYD, S., AND FALL, K. ns2 (network simulator 2). http://www-nrg.ee.lbl.gov/ns/.

[27] MERKLE, R. C. A digital signature based on a conventional encryption function. In *Proceedings of CRYPTO* (1987).

[28] MUÑOZ, J. L., FORNÉ, J., ESPARZA, O., AND REY, M. Efficient certificate revocation system implementation: Huffman merkle hash tree (HuffMHT). *Trust, Privacy and Security in Digital Business 3592* (2005), 119–127.

[29] PANNETRAT, A., AND MOLVA, R. Efficient multicast packet authentication. In *Proceedings of NDSS* (2003).

[30] PERRIG, A. The BiBa one-time signature and broadcast authentication protocol. In *Proceedings of ACM CCS* (2001).

[31] PERRIG, A., CANETTI, R., TYGAR, J. D., AND SONG, D. The TESLA broadcast authentication protocol. *RSA CryptoBytes* (2002).

[32] RAYA, M., AND HUBAUX, J.-P. Securing vehicular ad hoc networks. *JCS-SASN* (2007).

[33] REED, I., AND SOLOMON, G. Polynomial codes over certain finite fields. *J. SIAM 8*, 2 (1960), 300–304.

[34] RISTANOVIC, N., PAPADIMITRATOS, P., THEODORAKOPOULOS, G., HUBAUX, J.-P., AND LEBOUDEC, J.-Y. Adaptive message authentication for vehicular networks. In *Proceedings of ACM VANET* (2009).

[35] ROUF, I., MILLER, R., MUSTAFA, H., TAYLOR, T., OH, S., XU, W., GRUTESER, M., TRAPPE, W., AND SESKAR, I. Security and privacy vulnerabilities of in-car wireless networks: A tire pressure monitoring system case study. In *Proceedings of USENIX Security Symposium* (2010).

[36] SCHOCH, E., AND KARGL, F. On the efficiency of secure beaconing in VANETs. In *Proceedings of ACM WiSec* (2010).

[37] STUDER, A., BAI, F., BELLUR, B., AND PERRIG, A. Flexible, extensible, and efficient VANET authentication. *Journal of Communications and Networks 11*, 6 (Dec. 2009), 574–588.

[38] WANG, R., DU, W., AND NING, P. Containing denial-of-service attacks in broadcast authentication in sensor networks. In *Proceedings of ACM MobiHoc* (2007).