Authentication schemes for VANETs: a survey

Abhijit Das* and Dipanwita Roychoudhury

Department of Computer Science and Engineering,
Indian Institute of Technology Kharagpur,
West Bengal, India
Email: abhij@cse.iitkgp.ernet.in
Email: drc@cse.iitkgp.ernet.in
*Corresponding author

Debojyoti Bhattacharya, Rajavelu Srinivasan,
Rajeev Shorey and Tony Thomas

General Motors Technical Centre India,
India Science Lab,
Bangalore, Karnataka, India
Email: debojyoti.bhattacharya@gm.com
Email: srinivasan.rajavelu@gm.com
Email: rajeef.shorey@gm.com
Email: t.thomas@gm.com

Abstract: In this paper, we make a survey of the known schemes for V2V (vehicle to vehicle) and V2I (vehicle to infrastructure) authentication in VANETs (vehicular ad hoc networks). We consider a variety of safety applications in VANETs, identify authentication as one of the security requirements and discuss the security challenges for these applications. Existing authentication schemes based on digital signatures are discussed along with schemes based on hash chains and hash trees. It has also been shown how the level of security of these schemes is increased by the involvement of a trusted Certification Authority. We also focus on issues pertaining to anonymity, unlinkability, traceability and computation and communication overhead. Finally, we summarise the limitations of the existing authentication schemes in real-life applications and conclude that further research is essential in this area.

Keywords: VANET; safety; security; authentication; digital signature; hash chain; hash tree; public-key infrastructure; group communication.


Biographical notes: Abhijit Das received ME and PhD degrees in Computer Science and Engineering from Indian Institute of Science, Bangalore in the years 1993 and 2000, respectively. He is currently Associate Professor in the Department of Computer Science and Engineering, Indian Institute of Technology Kharagpur. He held academic positions at Indian Institute of
Technology Kanpur and Ruhr-Universität Bochum, Germany. His research interests include arithmetic and algebraic computations with specific applications to cryptology.

Dipanwita Roychoudhury received her BTech and MTech degrees in Computer Science from University of Kolkata in 1987 and 1989, respectively, and the PhD degree from the Department of Computer Sc. and Eng., Indian Institute of Technology Kharagpur in 1994. She is a Professor in the Department of Computer Science and Engineering, Indian Institute of Technology Kharagpur, India. Her current research interests are in the field of cryptography, error correcting code, cellular automata, and VLSI design and testing. She is the recipient of INSA Young Scientist Award and Associate of Indian Academy of Science and is the Fellow of Indian National Academy of Engineers (INAE).

Debojyoti Bhattacharya is currently a researcher in ABB India Corporate Research Center, where he is part of Industrial Communication group. His research interest is security protocols and applications. He received his MS from Computer Science Department of Indian Institute of Technology Kharagpur in the year 2008.

Rajavelu Srinivasan is currently working as a Regional Solution Manager in Nokia Siemens Network, Dubai. He worked as Innovation Manager focusing on Telco SaaS development in his past role with Nokia Siemens Networks R&D, Bangalore. He in his current role interacts closely with communication service providers in creating solutions in the area of IP Service Migration and Network Analytics for enhancing QoE/QoS. He worked with General Motors R&D, India Science Lab as Researcher contributing to the Vehicle-To-Vehicle Communications technology. He holds a PhD from the Faculty of Information and Communication Engineering from Anna University, Chennai. He has to his credit issued international patent and several publications in the area of information security.

Rajeev Shorey is the President of NIIT University, India, since August 2009. He received his PhD and MS (Eng.) in Electrical Communication Engineering from IISc, Bangalore, India, in 1997 and 1991, respectively, and BE degree in Computer Science and Engineering from IISc, Bangalore in 1987. Prior to joining NIIT University he was the Lab Group Manager of the Vehicle Communications and Information Management Group at GM ISL. He serves on the Editorial boards of IEEE Transactions on Mobile Computing and Wireless Networks, Journal of Mobile Communication, Computation and Information, and editor of the book titled Mobile, Wireless and Sensor Networks: Technology, Applications and Future Directions.

Tony Thomas received his MSc and PhD degrees in Mathematics from IIT Kanpur in 1998 and 2006, respectively. From Aug 2006 to July 2007, he was a Postdoctoral Researcher at KAIST, Korea. From Oct 2007 to Sep 2008 he worked as a Research Associate at the GM Research Lab, Bangalore. From Oct 2008 to Mar 2011 he was a Research Fellow at the School of Computer Engineering, NTU, Singapore. He is currently an Asst. Professor at Indian Institute of Information Technology and Management, Kerala, India. His research interests are in cryptography, DRM and information security.
1 Introduction

Vehicular Ad Hoc Networks (VANETs) form a special class of Mobile Ad Hoc Networks (MANETs) and have, of late, become an important area of academic and industrial research. Vehicles equipped with embedded processors and capable of communicating with each other by wireless broadcast mechanism are being manufactured. As a result, a whole new bunch of applications, geared to the needs and capability of these smart vehicles, is coming up. The most important objective is to heighten the safety of driving, for example to avoid accidents, to request for emergency help, to report crashes and so on. Cooperative driving also helps better utilisation of roads, such as traffic optimisation for avoidance of traffic jams. Other services like payment of toll taxes, automatic refuelling and infotainment (connection to the internet) can also be automated for smart vehicles.

Vehicles need to communicate with each other (V2V, vehicle to vehicle, communication) and with road-side infrastructure (V2I, vehicle to infrastructure, communication). Safety messages sent by vehicles can be classified into three categories (Raya and Hubaux, 2007). Periodic beacons are transmitted by vehicles at a regular interval of 10–100 ms in order to disseminate general traffic information to other vehicles and to the infrastructure. General safety-related messages are meant for cooperative driving and avoidance of accidents. Finally, liability messages pertain to liability-related incidents, like reporting crashes.

These messages are usually broadcast by vehicles and are not meant for node-to-node communication. Consequently, confidentiality of these messages is not important, that is issues like message encryption are not relevant for safety applications in VANETs. However, to guard against several forms of attacks made by insiders and outsiders, each vehicle needs to authenticate itself to the recipients of broadcast messages. Moreover, data integrity should be established in order to avoid active attacks like injection of bogus information and replay of messages.

Standardisation initiatives such as IEEE 1609.2 (IEEE, 2006), CAMP (NHTSA, 2005), Car 2 Car Communication Consortium (http://www.car-to-car.org/) and the DSRC (Dedicated Short Range Communications; http://www.standards.its.dot.gov/Documents/advisories/dsrc_advisory.htm) Consortium have been taken to address the problem of secure communication in VANETs. The NoW (Network on Wheels; http://www.network-on-wheels.de/) and the SeVeCom (Secure Vehicle Communication; http://www.sevecom.org/) projects are also worth mentioning in this context.

In the rest of this paper, we provide a survey of some recent schemes proposed for achieving authentication in VANETs. Our focus is on cryptographic techniques. Issues like intrusion detection (Leinmüller et al., 2004), identification of malicious nodes (Golle et al., 2004) and bogus information (Picconi et al., 2006) and determination of vehicle locations (Hubaux et al., 2004; Parno and Perrig, 2005) are not discussed here.

The paper is organised as follows. Section 2 reports some safety applications in VANETs. Issues like security requirements and challenges are identified in Section 3. Existing authentication mechanisms are discussed in Section 4, whereas Section 5 deals with Public-Key Infrastructure (PKI). Section 6 discusses security issues like anonymity, unlinkability and traceability. In Section 7, some alternate authentication models in VANETs are discussed. Finally, Section 8 concludes the paper after identifying some promising areas of future research.
Some safety applications in VANETs

Some applications where vehicles send safety-related messages are listed here. This list is based upon Bai et al. (2006).

1. **Stopped/slow vehicle advisor**: When a vehicle slows down or stops, it notifies its neighbouring vehicles about this event.
2. **Emergency braking notification**: A vehicle notifies its neighbours when it applies emergency brakes.
3. **Crash notification**: A vehicle that has encountered an accident broadcasts warning messages as long as the site of the accident is not clear.
4. **Road condition notification**: Bad road conditions (like presence of ice) are broadcast by an affected vehicle.
5. **Road feature notification**: Road features (like bends and hill) are broadcasted by a vehicle to its neighbours.
6. **Collision warning**: Tracking kinematic information (like speed and distance) from neighbouring vehicles helps a vehicle to raise potential collision alerts to itself and to its neighbours.
7. **Traffic violation warning**: Information periodically transmitted by Road-Side Units (RSUs) is used by a vehicle to issue warning messages about potential violation of traffic signals.

The authenticity of every safety-related message transmitted by a vehicle needs to be established to the neighbouring vehicles and/or to the RSUs. Non-safety applications, like traffic management and optimisation, non-stop toll collection and availability and choice of parking space, also demand authenticated message transmissions.

Security issues in VANETs

Before a discussion on VANET security protocols, it is necessary to identify the goals to achieve and the hurdles to overcome.

3.1 Security requirements

The five most important attributes of secure communication in VANETs are the following:

1. **Authentication**: As mentioned earlier, it is necessary for a vehicle to authenticate itself in every message it sends. In other words, an outsider should not be allowed to use the VANET protocol to send unauthorised messages to the network. It should also be infeasible to impersonate an authorised vehicle and send bogus messages on its behalf.
2. **Data integrity**: A message sent by a vehicle cannot be tampered by an active adversary without being detected by a receiver.
Privacy/anonymity: in a typical scenario, the identity of a vehicle is linked to the identity of its driver in a unique way. For the sake of privacy, a driver does not want his/her identity being disclosed to others (including local authorities and law enforcement agencies). If messages transmitted by a vehicle carry the identity of its driver, it becomes easy to analyse the recorded transcripts to encroach upon the privacy of the driver. It is to be noted that the license plate of a vehicle is already a breach of privacy. However, its scope is restricted to visual inspection from only a limited distance. Violation of privacy is generally accepted by the driving community only to this extent. A security protocol for VANETs may not leak any further information about the identity of a vehicle or its driver.

Unlinkability: An adversary should not be able to relate multiple messages sent by the same vehicle.

Traceability/non-repudiation: A sender should not be able to deny having sent a message. This is a requirement specifically for liability-related messages. While other vehicles, transport agencies and police may not link vehicle identities with stored messages; higher authorities (like the court) should have the capability to overrule the privacy requirements of a sender.

3.2 Challenges

A fair implementation of the above security requirements is by all means a challenging task. There are certain issues that make a VANET significantly different from a traditional MANET.

1 Contradictory security requirements: It is apparent that achieving authentication does not go well with the requirement of privacy. Privacy and unlinkability, in their turns, conflict with the notion of traceability. In a VANET, however, all the above security requirements have to be adequately addressed.

2 Resource-constrained computing: A modern vehicle comes with tens (even hundreds) of embedded processors catering to a variety of needs of the vehicle. In order to keep vehicle prices affordable, a manufacturer would not go for very high-end processors. A processing speed of only a few hundred MHz is a standard in most vehicles. Compare this with a Pentium processor (often multi-core) running at a clock speed of several thousand MHz. Moreover, a huge amount of memory may not be expected for each embedded processor in a vehicle. One has to be satisfied with only a few hundred kilobytes of memory. On the contrary, several applications (particularly, those related to safety, like collision avoidance) must run in real time. Long computations and massive storage overheads are not acceptable in these situations.

3 Low communication bandwidth: As vehicles communicate by broadcasting, network bandwidth may also be a bottleneck, particularly in heavily crowded roads.

4 Lossy channel: The vehicular wireless communication channel is lossy in nature. So there are chances that some broadcast authentication schemes (such as TESLA) fail if the anchor messages get lost in the channel.
Highly dynamic network topology: Nodes in a VANET are extremely mobile and may have high relative speeds. Moreover, vehicles have only a limited communication range (not exceeding 1000 m, but usually much less, around 100–300 m) and so the topology of a VANET changes rather rapidly. Two vehicles moving in opposite directions may meet only once. On the other hand, two vehicles moving in the same direction are expected to remain neighbours for some time. Finally, the mobility pattern in highways is markedly different from that at crossings.

Unavailability of infrastructure: Vehicles need to connect to the infrastructure using RSUs for different requirements, such as to acquire cryptographic credentials and report malicious activities. However, during the initial years of deployment, one may not expect RSUs to be accessible from everywhere. So a vehicle may be forced to carry out authenticated message transfer without or with infrequent availability of infrastructural help.

Scalability: A VANET may involve millions of vehicles. Security protocols for a VANET should support extremely huge networks.

Interoperability: A VANET consists of vehicles from several manufacturers. Moreover, local regulations may vary (possibly considerably) from place to place. Even different states in a country (as in the USA) may have variations in applicable law. A security protocol should take into account this varied vehicular and spatial characteristics.

Asymmetry in the number of signature generation to the number of verifications: In a VANET scenario, a vehicle has to verify all the messages from its neighbours, whereas it signs its own messages only. Therefore, verification has to be more efficient than signing. Typically, a vehicle has to perform in the order of ten signature verifications per signature generation.

A VANET has some mitigating features too.

Computing resources are not too scarce: Although vehicles are not equipped with plentiful computing resources, processor speed and memory are not too scarce (compare with sensor networks). Moreover, battery power is not inconveniently limited in a vehicle (compare with sensor and cellular networks). A good amount of computation can be carried out in real time by a vehicle’s processors. This issue is particularly relevant, since a full-fledged deployment of smart vehicles is expected to take a decade or longer (Raya and Hubaux, 2007). More able computing resources with lower costs will be available in near future.

Hardware support: Most of the present-day vehicles are equipped with a significant amount of hardware support such as EDR (Event Data Recorder, like the black box in an aeroplane), GPS (Global Positioning System) or DGPS (Differential GPS), making the VANET security goals somewhat easily achievable. Cryptographic data can be embedded in tamper-resistant devices like smart cards. Attempts to manhandle the devices would delete (zeroize) all sensitive information stored there. Dedicated crypto coprocessors (as recommended by Leinmüller et al., 2006, for instance) are a costlier, albeit affordable, alternative to smart cards.
Authentication schemes for VANETs

3 Information aggregation: In VANETs, one vehicle may receive the same message from multiple senders such as more than one vehicles reporting road features. This may help in designing computationally efficient algorithms with a weaker notion of authenticity, such as batch verification.

4 Physical laws: Kinematic information of vehicles and physical laws can be exploited to design efficient authentication mechanisms. For example, one vehicle cannot report in two different locations at the same time.

To sum up, we need to look at secure and moderately efficient authentication schemes that support anonymity, unlinkability and traceability.

4 Authentication mechanisms

In this section, we describe three basic mechanisms for achieving authentication in a VANET.

4.1 Authentication based on public-key cryptosystems

As long back as Arazi (1991) mentioned the possibility of using public-key cryptography in the form of digital signatures in order to achieve both authentication and message integrity. Subsequently, many researchers suggest public-key techniques for VANET applications, see for example the NoW recommendation (Gerlach et al., 2007) and the SeVeCom recommendations (Leinmüller et al., 2006; Papadimitratos et al., 2007). Digital signatures turn out to be the most widely suggested form of authentication mechanism in VANETs.

4.1.1 Digital signatures

In a public-key (or asymmetric) cryptosystem (Menezes et al., 1997) each user $U$ has two keys: the public key $PuK_U$ and the private key $PrK_U$. The public key is made public, whereas the private key is kept secret.

In order to digitally sign a message $M$, the user $U$ first computes a short representative $m = H(M)$ of $M$ using a hash function $H$. Subsequently, $U$ applies a signature generation operation to obtain $S_U = sgn(m, PrK_U)$. The pair $(M, S_U)$ is the message $M$ signed by $U$.

Anybody having access to the public key of $U$ can verify the authenticity of $U$’s signature on $M$ by checking whether the condition $H(M) = vrf(S_U, PuK_U)$ is satisfied.

The generation of the signature $S_U$ uses the private key of $U$. Since this key is kept secret, nobody is expected to generate $S_U$ from $M$ alone. On the other hand, verification of a signed message can be carried out by anybody, since it uses publicly available information only.

If a vehicle $U$ wants to send a message $M$, it first generates a digital signature $S_U$ on $M$ and broadcasts $(M, S_U$ and $PuK_U)$. Any other vehicle in the communication range of $U$ checks the validity of $U$’s signature by performing the verification step. This is how sender authentication is achieved. Data integrity follows from the properties of cryptographic hash functions (Menezes et al., 1997). It is considered difficult to obtain a
message $M'$ different from $M$ such that $H(M) = H(M')$, that is an adversary is expected to be unable to sign any message $M'$ only from one or more (but a moderate number of) signatures of $U$.

4.1.2 Choice of the digital signature algorithm

So far as the applicability in VANETs is of concern, a Digital Signature Algorithm (DSA) is assessed in terms of the following performance metrics:

1. **Signature generation time**: It is required to minimise the time for carrying out the operation $sgn$ to produce $S_U$ from $M$.

2. **Signature verification time**: Analogously, one aims at reducing the computational effort associated with the application of the operation $vrf$. Since each message is signed only once but verified by multiple recipients, or equivalently since a vehicle has to verify messages from all of its neighbours (whereas it signs its own messages only), it is more important to make the verification step efficient (compared to the signature generation step).

3. **Size of keys**: The public key is transmitted along with the signed message. On the other hand, the private key needs to be stored secretly in a vehicle’s tamper-resistant device. In order to enhance communication and storage efficiency, small keys are preferred.

4. **Size of signatures**: The signature $S_U$ is broadcast and should, therefore, be as small as possible.

A host of DSAs has been proposed in the literature. Among these, the following three algorithms appear to be the most suitable for VANETs:

1. **RSA** (Rivest et al., 1978): The oldest known asymmetric signature algorithm, RSA, enjoys good security properties. It has withstood attacks for over three decades. A property of the RSA algorithm is that very small (single-precision) public keys can be chosen making the verification process quite efficient. In view of this, El Zarki et al. (2002) recommend the use of RSA. However, the length of RSA signatures is rather large.

2. **NTRUSign** (Hoffstein et al., 2003): The biggest advantage of the NTRU signature algorithm is that it is significantly faster than other DSAs. Its key size and signature size are somewhere between those for RSA and ECDSA. However, several attacks on NTRUSign are known (see Gentry and Szydlo, 2002; Nguyen, 2006). In view of these attacks, the security of NTRUSign remains questionable.

3. **ECDSA** (Johnson and Menezes, 1999): The elliptic curve version of the DSA corresponds to small key and signature lengths. Several efficient software and hardware implementations of the elliptic curve arithmetic are known, for example Hankerson et al. (2004).

The above signature algorithms are implemented and compared by Dötzter et al. (2005), Leinmüller et al. (2004) and Raya and Hubaux (2007). SeVeCom (Papadimitratos et al., 2007) recommends the use of ECDSA.
4.2 Authentication based on hash chains

An asymmetric algorithm is inherently slower than a symmetric algorithm (by a factor of about 1000). Keyed hash functions (like HMACs; Bellare et al., 1996; IETF, 1997) can be used to achieve both sender authentication and data integrity. However, a symmetric algorithm requires pairwise key establishment among communicating parties and the resulting overhead turns out to be massive.

4.2.1 TESLA

Perrig et al. (2002) propose a broadcast authentication protocol TESLA (Timed Efficient Stream Loss-tolerant Authentication), which uses a clever combination of symmetric and asymmetric techniques in order to arrive at an efficient authentication scheme for wireless networks. Hu and Laberteaux (2006) discuss the adaptation of TESLA to VANETs.

TESLA uses symmetric techniques in order to sign a message and to verify a signature. The signing key itself is disclosed when its period of validity expires. In that sense, time imposes an asymmetry in a TESLA-based scheme. TESLA certificates (see Section 5.2) are, however, based on public-key techniques.

TESLA uses a one-way function $F$ for generating a chain of keys $K_0, K_1, ..., K_N$. To start with, a random key $K_N$ of the desired length is created. Subsequently, for $i = N - 1, N - 2, ..., 0$ (in that sequence), one generates the key $K_i = F(K_{i+1})$. Typically, a cryptographic hash function is used to realise $F$. Under cryptographic assumptions, it is difficult to obtain the key $K_{i+1}$ from a knowledge of $K_0, K_1, ..., K_i$ only.

Use of the TESLA key chain should also specify a duration $\Delta$ of validity for each key $K_i$ and also the time instant $T_0$ at which the use of the chain begins. The keys are to be used in the reverse order as they are generated, that is in the order $K_1, K_2, ..., K_N$. The key $K_i$ remains valid during the time interval $[T_0, T_0 + \Delta]$, $K_2$ during $[T_0 + \Delta, T_0 + 2\Delta]$ and so on. More generally, the interval of validity for the key $K_i$ is $[T_0 + (i - 1) \Delta, T_0 + i\Delta]$.

After the key $K_i$ expires (at time $T_0 + i\Delta$), the owner of the key chain discloses $K_i$ by broadcasting it in plain text. The key $K_0$ is revealed at $T_0$ itself; it is not used for signing any message, but for proving the authenticity of the other keys in the chain. $K_0$ is called the anchor of the chain.

**Figure 1** TESLA key chain
TESLA-based authentication assumes that the nodes in the network have synchronised clocks. Use of GPS-enabled clock settings and crystal oscillators (which reside in tamper-resistant hardware so as to preclude unauthorised alterations) allows vehicles to maintain time with an accuracy less than 100 $\mu$s (Hu and Laberteaux, 2006). The period $\Delta$ for each key is, on the other hand, of the order of 10 ms. Nonetheless, an error $\epsilon$ (like 100 $\mu$s) needs to be incorporated during the validation of the interval of a key.

During the $i$-th interval $[T_0 + (i - 1)\Delta, T_0 + i\Delta]$, the signer uses the key $K_i$ and signs the message $M$ as:

$$S = \text{HMAC}_{K_i}(M),$$

where HMAC is a keyed hash function (Bellare et al., 1996; IETF, 1997). The signed message is the pair $(M, S)$.

Upon reception of $(M, S)$, a verifier waits until the period of validity of $K_i$ expires and the signer broadcasts $K_i$. If the time of receiving $(M, S)$ is not in the interval $[T_0 + (i - 1)\Delta - 2\epsilon, T_0 + i\Delta + 2\epsilon]$, the message $M$ is discarded. Otherwise, the verifier checks whether $S$ equals $\text{HMAC}_{K_i}(M)$.

In order to verify that $K_i$ is a valid key of the signer, a verifier checks whether $K_0 = F(K_i) = F(F, \ldots, (F(K_i), \ldots))$ ($i$-fold application of $F$ on $K_i$). This verification can be made incremental, that is if $K_i$ is already verified as valid, then one can validate $K_{i+1}$ by verifying whether $K_i = F(K_{i+1})$. Since $F$ is a one-way function, it is assumed that nobody can generate a future key from the knowledge of the disclosed keys only.

The biggest advantage of TESLA is its efficient mechanism for signature generation and verification. An HMAC typically involves only a symmetric cipher or a hash function and is, therefore, about three orders of magnitude faster than its asymmetric equivalent. Validation of $K_0$ uses asymmetric techniques, but subsequent validations of $K_1, K_2, \ldots, K_N$ require applications of $F$ only. Since $F$ is typically realised by a hash function, it is also quite fast. Moreover, since each $K_i$ is valid for only a short period (like 10 ms) and is publicly disclosed after that, it is not necessary to guard $K_i$ with a long-term security. Consequently, one may take each $K_i$ as a short key (64–80 bits only) and this further enhances computational and storage efficiency.

On the darker side, a TESLA-based scheme suffers from the problem of deferred authentication. Until $K_i$ is revealed, no receiver can verify signatures made by $K_i$. In safety-critical and/or real-time situations, this delay may be unacceptable.

### 4.2.2 BiBa

In TESLA, *time* poses the basic asymmetry between a signer and a verifier. Perrig’s Bins-and-Balls (BiBa) signature scheme (Perrig, 2001) is another authentication scheme based upon hash chains and achieves the necessary asymmetry using the *birthday paradox*. BiBa attempts to decrease the signature size and verification complexity, but the public keys are quite large and the time to generate signatures is high at the signer side. However, as the signature size is small and the verification is efficient, BiBa can be used to design new protocols for broadcast authentication in VANETs. It has a probabilistic nature that enables the reuse of same keys for several signatures, albeit with a security degradation for each additional key reuse.
BiBa uses multiple hash chains with keys $K_{i,j}$ for $i = 1, 2, \ldots, t$ and $j = 0, 1, 2, \ldots, N$. Each $K_{i,j}$ is called a SEAL (Self-authenticating value). SEALs are disclosed in signatures leading to the possibility of immediate verification.

During the $j$-th interval $[T_0 + (j - 1) \Delta, T_0 + j \Delta]$, the SEALs $K_{1,j}, K_{2,j}, \ldots, K_{t,j}$ are used. In order to sign a message $M$, the sender first computes the hash $m = H(M)$. The sender then uses a pseudorandom function $G_m$ to generate the integers $G_m(K_{i,j})$ in the range $[0, R - 1]$ for $i = 1, 2, \ldots, t$. Finally, the sender locates a collision $G_m(K_{i,j}) = G_m(K_{i',j})$ and broadcasts the signed message $(M, K_{i,j}, K_{i',j})$. In order to verify this signature, one checks whether $K_{i,j} \neq K_{i',j}$ and $G_m(K_{i,j}) = G_m(K_{i',j})$, where $m = H(M)$.

**Figure 2** Collision in BiBa

If the signer fails to obtain a collision, (s)he tries with $m = H(M|c)$ for $c = 0, 1, 2, \ldots$. The value of $c$, for which a collision is obtained, should now be included in the signature.

Let us now review the security of BiBa. First, note that future SEALs cannot be predicted, since they are based on hash chains. The signer knows many ($t$) SEALs and can compute collisions $G_m(K_{i,j}) = G_m(K_{i',j})$ with high probability, whereas other entities know only a few SEALs from broadcasted signatures and cannot compute collisions easily. More precisely, the probability of collisions from a knowledge of $\tau$ SEALs is $P_c \approx 1 - e^{-\tau^2 R / 2}$. By the birthday paradox, $P_c \geq 0.632$ for $\tau \geq \sqrt{2R}$.

As an example, let $R = 757,000$ and $t = 1024$. The signer knows $\tau = 1024$ SEALs, so $P_c \approx 0.5$. An adversary knows (at most) $\tau = 2r$ SEALs from $r$ broadcasted signatures. For $r = 1$, $P_c \approx 2^{-17.94}$ and for $r = 4$, $P_c \approx 2^{-14.36}$.

Multi-way collisions increase the security of BiBa considerably. After hashing the message $m = H(M|c)$ (for $c = 0, 1, 2, \ldots$), the signer finds indices $i_1, i_2, \ldots, i_k$ with $G_m(K_{i_1,c}) = G_m(K_{i_2,c}) = \ldots = G_m(K_{i_k,c}) \in [0, R - 1]$. The signature on $M$ is $(K_{i_1,c}, K_{i_2,c}, \ldots, K_{i_k,c}, c)$.

For example, take $t = 1024$, $k = 16$ and $R = 168$. The signer can find a $k$-way collision with probability 0.5. An adversary, from a knowledge of only $4k$ SEALs (from four broadcasted signatures), can find a $k$-way collision with probability $2^{-58}$. 
4.2.3 HORS

The HORS scheme (Hash to Obtain Random Subset) proposed by Reyzin and Reyzin (2002) is an improvement of BiBa with \( k \)-way collisions. Compared to BiBa, HORS achieves significantly faster signature generation, slightly faster signature verification and slightly better security.

HORS is based upon subset constructions (instead of collisions, as in BiBa). Let \( L \) be the length of hash values (e.g., \( L = 160 \) for SHA-1). Let \( t \) be the number of hash chains used. We denote \( l = \log_2 t \) and write \( L = kl \).

In order to sign a message \( M \), the signer hashes the message to obtain \( m = H(M) \) and breaks \( m \) into \( k \) bit-strings \( m_1, m_2, \ldots, m_k \) each of length \( l \):

\[
    m = m_1 || m_2 || \ldots || m_k.
\]

Each \( m_i \) is treated as an integer \( i_u \) in the binary representation. The signature on the message \( M \) (in the \( j \)-th time interval) is \( \{K_{i_1,j}, K_{i_2,j}, \ldots, K_{i_t,j}\} \).

For assessing the security of HORS, we view

\[
    H(M) = \{i_1+1, i_2+1, \ldots, i_t+1\} \subseteq \{1, 2, \ldots, t\}.
\]

Therefore, the challenge posed to an adversary is: given signatures on \( M_1, M_2, \ldots, M_r \), to find a new message \( M_{r+1} \) such that \( H(M_{r+1}) \subseteq H(M_1) \cup H(M_2) \cup \ldots \cup H(M_r) \). \( H \) is called subset-resilient if this is difficult for small values of \( r \).

For instance, take the parameters \( L = 160 \) (SHA-1), \( k = 16 \), \( t = 1024 \) and \( l = 10 \). For \( r = 1 \), the forging probability is \( 2^{-96} \), whereas for \( r = 4 \), the forging probability is \( 2^{-64} \).

4.3 Authentication based on hash trees

One-time signatures offer an attractive alternative to public-key-based and hash-chain-based signatures, since they are based only on one-way functions. They are particularly suited to resource-constrained devices, as they do not involve costly arithmetic operations (Dods et al., 2005).

A one-time signature can be used to sign at most one message, otherwise signatures can be forged. Consequently, the scheme tends to be unwieldy when used to authenticate multiple messages, because additional keys need to be generated to both sign and verify each new message. By contrast, in a public-key signature scheme like RSA, the same key pair can be used to authenticate multiple documents. The public information necessary to verify one-time signatures is often referred to as validation parameters. When one-time signatures are combined with techniques for authenticating validation parameters, multiple signatures are possible.

4.3.1 Merkle’s one-time signature

The best-known one-time signature scheme was presented by Merkle (1987, 1989) and is an improvement of Lamport’s signature scheme (Lamport, 1979). Let \( H \) be a hash function (like SHA-1) which produces \( l \)-bit outputs. Denote \( t = l + \lceil \log_2 t \rceil \). Key generation in Merkle’s scheme involves the following steps to be executed by the signer.

1. Generate \( t \) random secret bit-strings \( x_1, x_2, \ldots, x_t \). The values \( X = (x_1, x_2, \ldots, x_t) \) are signing (or private) keys.

2. Compute \( y_i = H(x_i) \) for \( i = 1, 2, \ldots, t \). The values \( Y = (y_1, y_2, \ldots, y_t) \) are commitments (or public) keys.
Authentication schemes for VANETs

For now, we assume that the commitments are made available to the verifiers in an authenticated way.

- **Signature generation:** Upon input a message $M$, the following steps are carried out.
  - Hash the message $M$ to generate $m = H(M)$.
  - Calculate $m' = m \parallel C = m'_1m'_2\ldots m'_t$, where $C$ is the count of zero-bits in $m$.
  - As the signature, output all the secrets $x_i$ for which $m'_i$.

- **Signature verification:**
  - Calculate the checksum $C$ from $m = H(M)$ and obtain $m' = m \parallel C$.
  - Check whether exactly the appropriate secrets $x_i$ are revealed.
  - Check whether hashing each secret $x_i$ generates its respective claimed commitment $y_i$.

When an adversary pretends that a one-bit was actually a zero-bit, (s)he must also increase the value of the checksum. Moreover, the adversary must reveal the appropriate signing keys. Thus, the security of Merkle’s scheme is based upon the one-way property of $H$ (generation of $y_i$ from $x_i$) and the collision resistance of $H$ (generation of $m = H(M)$).

#### 4.3.2 Winternitz one-time signature

Winternitz signatures (Merkle, 1987; Merkle, 1989) provide a time-memory tradeoff. In this scheme, the message is processed in blocks of $w$ bits. Larger values of $w$ imply reduction in signature sizes and increase in signature generation and verification times (compared to the Merkle scheme). The values $w = 2, 3$ and 4 are typically recommended.

Let $H$ be a hash function (like SHA-1), and $l$ the bit length of outputs of $H$. Denote $H^i(x) = H(H(\cdots H(H(x))\cdots))$ ($i$-fold application of $H$), and $t = \left\lceil \log_2 \frac{w}{l} \right\rceil + \left\lceil \log_2 \frac{l}{w} \right\rceil + 1$.

Winternitz signing (private) keys are $t$ random bit-strings $X = (x_1, x_2, \ldots, x_t)$. The corresponding commitment (public key) is $Y = H(y_1, H(y_2), \ldots, H(y_t))$, where $y_i = H^{2^{i-1}}(x_i)$ for $i = 1, 2, \ldots, t$. We assume that the commitment $Y$ is supplied to the verifier in an authentic way.

- **Signature generation:**
  - Hash the message $M$ to generate $m = H(M)$.
  - Break $m$ into blocks $b_1, b_2, \ldots, b_{\lceil l/w \rceil}$ each of length $w$.
  - Compute the checksum $C = \sum_{i=1}^{\lceil l/w \rceil} 2^w - b_i$.
  - Break $C$ into $\lceil \log_2 \frac{l}{w} \rceil + 1 + w$ blocks $b_{\lceil l/w \rceil + 1}, \ldots, b_t$ each of length $w$.
  - Let $m' = m \parallel C = b_1b_2\ldots b_{\lceil l/w \rceil}h_{\lceil l/w \rceil + 1}\ldots h_t$.
  - For $i = 1, 2, \ldots, t$, compute $\sigma_i = H^b(x_i)$, where each bit-string $b_i$ is treated as an integer in the natural way.
  - The signature on $M$ is $(\sigma_1, \sigma_2, \ldots, \sigma_t)$. 
• **Signature verification:**
  
  - Compute $b_1, b_2, \ldots, b_t$ as during signature generation.
  - Compute $\phi_i = H^{2^{t-1-k}}(\sigma_i)$ for $i = 1, 2, \ldots, t$.
  - Compute $\Phi = H(\phi_1 \| \phi_2 \| \cdots \| \phi_t)$.
  - Check whether $\Phi = Y$.

4.3.3 **Merkle’s hash tree**

Merkle (1987) presents an infinite-tree signature which theoretically enables the creation of an unlimited number of one-time signatures. This scheme is impractical as the signature size is very large and grows logarithmically with the number of signatures. In a later paper, Merkle (1989) introduces the concept of hash trees to facilitate simultaneous authentication of public commitments of multiple one-time signatures. A Merkle hash tree is constructed as follows (see Figure 3).

- Construct a complete binary tree of height $h$.
- Generate $N = 2^h$ one-time key pairs $(X_1, Y_1), (X_2, Y_2), \ldots, (X_N, Y_N)$.
- Let $Z_i = H(Y_i)$ for $i = 1, 2, \ldots, N$.
- The leaves of the tree are labelled by $Z_1, Z_2, \ldots, Z_N$.
- An intermediate node has label:
  $\text{label}_{\text{left child}} \| \text{label}_{\text{right child}}$,
  where $H$ is a one-way hash function.
- The label at the root authenticates all of the commitments:
  $Y_1, Y_2, \ldots, Y_N$.

**Figure 3** A Merkle tree of height $h = 3$
A Merkle tree of height $h$ calls for a storage proportional to $N = 2^h$ and results in signatures of length proportional to $h$.

The one-time keys $(X_i, Y_i)$ are used in the sequence $i = 1, 2, \ldots, N$. In order to authenticate $Y_i$, the signer locates the unique path from the root to the leaf $Z_i$. The secret $X_i$, along with the labels of the siblings of all the nodes on this authentication path, constitute the signature (see Figure 4). From this information, a verifier can compute the label of the root, which is used to authenticate the message. One-time signatures based upon Merkle trees derive their security from the one-way property of the hash function $H$.

**Figure 4** Authentication of $Y_5$ in a Merkle tree

4.3.4 Fractal Merkle trees

Fractal Merkle trees proposed by Jakobsson et al. (2003) aim at reducing the storage requirements of Merkle trees in the signer’s machine. Let $h$ be the height of the Merkle tree and $\eta$ a divisor of $h$. Write $h = \eta \lambda$. The signer stores only $\lambda$ subtrees each of height $\eta$. The stored trees contain the current authentication path. For the next signature, some of the stored subtrees may need to be refreshed. The refreshing task can be distributed across different signature generations and has been shown to have a low amortised cost. Figure 5 shows a fractal Merkle tree with $h = 4$ and $\eta = \lambda = 2$.

**Figure 5** A fractal Merkle tree
4.3.5 FMTSeq

Naor et al. (2005) proposed FMTSeq (Fractal Merkle Tree Sequential Signatures) that involves seeding one-time secrets. Let \((X_i, Y_i)\) be the \(i\)-th one-time key pair, where \(X_i = (x_{i,1}, x_{i,2}, \ldots, x_{i,t})\) and \(Y_i = (y_{i,1}, y_{i,2}, \ldots, y_{i,t})\). Let \(F\) be a pseudorandom function. The signer uses a master secret key \(x\) and generates \(x_{i,j} = F(x, i, j)\) and \(y_{i,j} = H(x_{i,j})\).

Now, there is no need to store all \(x_{i,j}\) values. It is necessary to store only \(x\). The required keys \(x_{i,j}\) and the required labels of the nodes in the Merkle tree are computed during signing.

Naor et al. (2005) reported an efficient implementation of their seeding strategy on fractal Merkle trees. Table 1 is based upon their documented results. For about one million one-time signatures, using FMTSeq gives a fivefold improvement in the verification time and an improvement in the signing time by a factor of about 35, compared with 2048-bit RSA. Moreover, verification in FMTSeq with SHA-1 is three times faster and signing is about ten times faster than 1024-bit RSA.

<table>
<thead>
<tr>
<th>(H)</th>
<th>(h)</th>
<th>(\eta)</th>
<th>(\lambda)</th>
<th>Number of signatures</th>
<th>Signature time ((\mu s))</th>
<th>Verification time ((\mu s))</th>
<th>Memory (bytes)</th>
<th>Signature size (bytes)</th>
</tr>
</thead>
<tbody>
<tr>
<td>SHA-1</td>
<td>10</td>
<td>2</td>
<td>5</td>
<td>1024</td>
<td>757</td>
<td>73</td>
<td>1072</td>
<td>2864</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>5</td>
<td>2</td>
<td></td>
<td>1024</td>
<td>311</td>
<td>73</td>
<td>2156</td>
<td></td>
</tr>
<tr>
<td>SHA-256</td>
<td>10</td>
<td>2</td>
<td>5</td>
<td>1024</td>
<td>1184</td>
<td>111</td>
<td>1072</td>
<td>4400</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>5</td>
<td>2</td>
<td></td>
<td>1024</td>
<td>481</td>
<td>111</td>
<td>2156</td>
<td></td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>10</td>
<td></td>
<td>1503</td>
<td>79</td>
<td>79</td>
<td>3062</td>
<td></td>
</tr>
<tr>
<td>SHA-1</td>
<td>20</td>
<td>4</td>
<td>5</td>
<td>1048576</td>
<td>762</td>
<td>79</td>
<td>3352</td>
<td>3024</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>5</td>
<td>4</td>
<td></td>
<td>1048576</td>
<td>614</td>
<td>79</td>
<td>4690</td>
<td></td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>10</td>
<td></td>
<td>1048576</td>
<td>2358</td>
<td>114</td>
<td>3062</td>
<td></td>
</tr>
<tr>
<td>SHA-256</td>
<td>20</td>
<td>4</td>
<td>5</td>
<td>1048576</td>
<td>1188</td>
<td>113</td>
<td>3352</td>
<td>4560</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>5</td>
<td>4</td>
<td></td>
<td>1048576</td>
<td>954</td>
<td>120</td>
<td>4690</td>
<td></td>
</tr>
<tr>
<td>RSA-1024</td>
<td>20</td>
<td>4</td>
<td>5</td>
<td>1048576</td>
<td>5900</td>
<td>240</td>
<td>128</td>
<td></td>
</tr>
<tr>
<td>RSA-2048</td>
<td></td>
<td></td>
<td></td>
<td>33800</td>
<td>600</td>
<td>256</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Source: Naor et al. (2005)

4.3.6 Hierarchically ordered Merkle trees

Jakobsson and Wetzel (2004) proposed a family of solutions to the problem of providing attribute certificates with low generation and verification costs. These solutions are based on hash graphs and try to reduce the cost of Certificate Revocation Lists (CRL) (see Section 5.5). A number of solutions are given based on the hash graphs of different types like hash chains, Merkle trees and spatial Merkle trees. Jakobsson and Wetzel proposed an authentication scheme based on hierarchically ordered Merkle trees that is well suited to VANETs. The algorithm is shown in Figure 6.
Authentication schemes for VANETs

Figure 6  Jakobsson’s hierarchically ordered tree

A set of four Merkle trees is defined on a complete binary tree, where the leaves of each tree, that is the tree at each level of the main tree (Figure 6), are associated with different parameters of the VANET.

1 The fifth level comprises certified recommendations which are secret values until disclosed. The leftmost nodes on level five are independent random numbers from an interval large enough to preclude possibilities of guessing. Each other node in this level is the hash value of the left neighbouring node.

2 The trees at the fourth level are used to represent different access points within one geographical neighbourhood. The value of each leaf at this level is a hash function of the value of the rightmost node at Level 5. The hash values of the other nodes are derived from their children. In real-life scenario, instead of computing the hash value based on the value of the child, the hash can be computed also including the identity of the access point.

3 The trees at the third level correspond to geographical location; the leaves correspond to geographic neighbourhoods.

4 The Merkle trees at the second level correspond to time.

5 The Merkle tree at the top level is related to user clustering. The root value is associated with the CA (see Section 5) and the leaves correspond to different clusters of user performances; some of them may be application-specific. The root value is constant and public and assumed to be known by everyone. Thus, value of the root is a function of all leftmost Level 5 values.

Jakobsson and Wetzel (2004) studied on the cost of implementing the above signature scheme for VANETs. They opined that the verifier’s cost can be reduced if the ordering is properly done in the Merkle tree construction.
4.3.7 CMSS

Buchmann et al. (2006) proposed a variant of the Merkle signature scheme, called CMSS, to achieve reduced sizes of private keys and reduced times for generating key pairs and signatures. The authors develop a Java-based implementation and present several experimental evidences based upon signatures on 2^40 documents. They show that the key-pair generation time is reasonable. The signature generation and verification times are competitive or even superior to RSA and ECDSA. For authentication in VANETs, the main emphasis is to reduce the signing and verification times with limited memory storage. Consequently, CMSS or its variants may be a solution for VANETs.

5 Public-key infrastructure

Authentication mechanisms described in Section 4 are based upon the assumption that the signing and verifying keys are authentic. The desired level of confidence in this regard is achieved by the involvement of a trusted authority, called the Certification Authority (CA). The CA is typically a government agency or one approved by the government.

5.1 Public-key certificates

The authenticity of a digital signature is based upon the assumption that the corresponding verification (public) key is authentic. Without proper safeguards, nothing prevents an unauthorised attacker from preparing a key pair of his/her own choice and digitally signing messages with that private key.

Public-key certificates (IETF, 2008) are used to certify the genuineness of public keys. A public-key certificate contains a public key along with the credentials of the owner of the key. A certificate typically has a limited lifetime. The expiry date is also stored in the certificate. The authenticity of the certificate is guaranteed by the digital signature of the CA on the certificate.

During message broadcasting in a VANET, a vehicle $U$ sends $M$, $S_U$ and $Cert_U$ (instead of $M$, $S_U$ and $PuK_U$ as mentioned earlier). A recipient retrieves the public key $PuK_U$ from the certificate $Cert_U$ and checks the validity of $PuK_U$ by verifying the signature of the CA. In case CA’s signature is verified and the certificate has not expired, the recipient gains the desired confidence in using $PuK_U$ for verifying the signed message of $U$.

Since CAs are region-specific, digital certificates too vary from place to place. If a vehicle wants to communicate in a region not under the jurisdiction of its own CA, it is required to cross-certify the local CA with the vehicle’s CA.

5.2 TESLA certificates

Authenticity of any key $K_i$ in the protocol reduces to the authenticity of $K_0$. It, therefore, suffices to certify $K_0$ only. In order to authenticate $K_0$, the signer uses an asymmetric algorithm, that is $K_0$ is signed by the private key of the signer and the signature is to be verified by the corresponding public key. This public key, in its turn, is certified as authentic by the CA as in a usual PKI. To sum up, a TESLA certificate consists of the identity of the user, its public key, the signature of the CA on the public key, TESLA
Authentication schemes for VANETs

parameters ($N$, $T_0$, and $A$), the initial key $K_0$ and the signature of the user on $K_0$. This implies that a TESLA certificate is somewhat larger than a conventional PKI certificate. The large size of a TESLA certificate is, in practice, not a huge problem, since it is used relatively infrequently (in order to validate $K_0$ only).

5.3 BiBa certificates

BiBa is based on $t$ TESLA key chains. Only the anchors need to be certified. Therefore, a BiBa certificate consists of the parameters $N$, $T_0$, and $\Delta$; the anchors $K_{1,0}, K_{2,0}, \ldots, K_{t,0}$; the signatures of the user on these anchors; and the signature of the CA on the public key of the user. The HORS scheme admits identical certificates.

5.4 Certificates for one-time signatures

For a one-time signature $(X, Y)$, the public commitment $Y$ needs to be certified. For multiple one-time signatures based upon hash trees, the commitment $Y$ at the root needs to be certified. One may obtain CA’s signature on $Y$, or one may sign $Y$ with his/her private key and obtain the certificate of the CA on the corresponding public key.

5.5 Certificate revocation

Certificates are often revoked by the CA, for example if the private key corresponding to a certificate is compromised (or suspected to be so), or if the user is observed to have been doing malicious activities. A revoked certificate may not be used to derive the authenticity of a public key.

The CA maintains a list of revoked certificates. The list is called the CRL. A verifier needs access to the CRL in order to ensure that the certificate received in a broadcasted message has not been revoked by the CA. Two issues related to CRLs are addressed here.

5.5.1 Storage of the CRL

In a large network, the number of revoked nodes may be large. Under pseudonymous authentication (see Section 6), each vehicle possesses multiple certificates. This means that individual certificates (rather than individual users) are revoked, leading to potentially large storage overhead for the CRL in each vehicle.

This overhead may be reduced by storing a compressed representation of the CRL. Raya et al. (2006b) advocate the use of Bloom filters to that effect. A Bloom filter (Bloom, 1970) is a probabilistic data structure for testing the membership of an element in a set. If the Bloom filter outputs no, the element is certainly not present in the set. If it outputs yes, then there remains a chance that the element is actually not present in the set. However, the probability of these false positives can be configured to an acceptably small value (at the cost of increased storage and computation overhead). Raya et al. (2006b) estimate a size of few tens of kilobytes for the storage of the CRL.

Another approach suggested by Raya et al. (2006b, 2006c) is the use of short-lived certificates. Each certificate has only a small period of validity (like a day) and is never revoked (it automatically expires after the period of validity). Using short-lived certificates calls for frequent interaction with the infrastructure for periodic renewal of certificates. Moreover, absence of the revocation mechanism leaves a small window of vulnerability.
5.5.2 Distribution of the CRL

Only the CA has the authority to (issue and) revoke certificates and maintain the CRL. The CRL changes with time upon revocation of new certificates. A vehicle needs to establish a contact with the CA for obtaining the latest version of the CRL. A delta-CRL stores only the differences between two versions of the CRL and is expected to be significantly smaller in size than actual CRLs. A vehicle may update its currently saved CRL by obtaining a delta-CRL from the CA.

CRL updating is not at all possible in a region having no infrastructural support. Raya et al. (2006b) proposed a Distributed Revocation Protocol (DRP) for temporarily disabling certificates belonging to suspected attackers and also for notifying the CA about suspected malicious activities. Only the CA is empowered to formally revoke certificates. The DRP is meant to issue warning messages to neighbouring vehicles in regions lacking contact with the CA.

Revocation of the tamper-proof security devices belonging to a vehicle is also suggested (Raya et al., 2006b) as a method of revocation of all certificates residing in the vehicle.

5.6 Micali certificates

In order to reduce computation and communication overheads associated with CRLs, Micali (1997) proposed a modification called Certificate Revocation Status (CRS). Micali assumes that a certificate is valid for a year, and revocation information is disseminated by the CA on a daily basis. In addition to the usual components, Micali’s certificate contains two short strings $Y$ and $N$ generated as follows. The CA randomly chooses two secret strings $Y_0$ and $N_0$, computes $Y_i = H(Y_0)$ for $i = 1, 2, \ldots, 365$ and takes $Y = Y_{365}$ and $N = H(N_0)$. Here, $H$ is a one-way hash function (like SHA-1). The string $Y_{365-i}$ is a short proof of the validity of the certificate on the $i$-th day. Upon request from a verifier, the CA sends either $N_0$ or $Y_{365-i}$, depending upon whether the certificate is revoked or not. Let $Z$ be the string received from the CA. If $H(Z) = N$, then the verifier concludes that the certificate is revoked. If $H(Z) = Y$, then the verifier concludes that the certificate continues to be valid on the $i$-th day. Finally, if neither of these two conditions holds, the verifier concludes that the CA is denying service.

Micali’s method is evidently less expensive than CRLs, but traversing hash chains is again computationally costly. Later, efficient methods (e.g. Coppersmith and Jakobsson, 2002; Sella, 2003) are proposed using hash chains.

6 Anonymity, unlinkability and traceability

A digital certificate contains some credentials of the owner of the signing key. Transmitting these information in a message amounts to violation of the privacy of the sender and is unacceptable to many drivers. This issue can be resolved by using anonymous (also called pseudonymous) certificates.

6.1 Use of pseudonyms

Instead of using the CA’s original certificate, a vehicle may obtain and transmit temporary certificates containing no direct information about the identity of the vehicle. Anonymity in a VANET is thus achieved in the form of pseudonymity.
6.2 Reanonymisation

Using pseudonyms achieves anonymity, but does not meet the requirement of unlinkability. It remains possible to track a vehicle by tracking messages carrying the pseudonym of the vehicle.

The concept of reanonymisation (refreshing pseudonyms) is proposed by Parno and Perrig (2005). The idea is to keep on changing pseudonyms so as to frustrate attempts to track specific pseudonyms. Hubaux et al. (2004) proposed a measure of anonymity based on entropy. Calandriello et al. (2007) estimated, under a reasonable model of the network, that it suffices for a vehicle to use no more than 200,000 pseudonyms per year. They take one minute as the lifetime of each pseudonym. A pseudonym has to be discarded after this period and may not be reused later.

Gerlach (2006) makes a risk analysis for VANETs and indicates that changing pseudonyms only cannot prevent several attacks. He proposes the use of context-sensitive information in order to identify opportunistic moments for pseudonym changes.

Each pseudonym must bear the signature of the CA. It is, therefore, necessary for a vehicle to contact the CA and refresh its stored list of pseudonyms (after authenticating itself to the CA possibly using an old certified pseudonym). Doing this once a year requires a storage overhead of about 23 megabytes (for ECDSA). If pseudonym refuelling stations are available frequently (such as in toll tax collection centres), then all these pseudonyms need not be downloaded at once.

The concept of self-generated pseudonyms is introduced by Calandriello et al. (2007). The CA needs to be contacted for obtaining the certificates for the pseudonyms. Rahman and Hengartner (2007) proposed a concrete realisation of self-generating pseudonyms. They use a blind signature protocol based upon the BLS short signature scheme (Boneh et al., 2004).

Kamat et al. (2006) proposed an identity-based framework for the generation of new pseudonyms. They use the signcryption scheme of Chen and Malone-Lee (2005). A vehicle desiring to obtain a new pseudonym sends its signcrypted credentials (including its permanent certificate and current pseudonym) to a base station. After verifying the credentials, the base station generates a new pseudonym based on the current timestamp and the permanent identity of the vehicle. The pseudonym is protected by the secret AES key of the base station and is communicated to the vehicle in encrypted form (say, by RSA).

Armknecht et al. (2007) proposed a method for a completely autonomous generation of pseudonyms. During the set-up phase, a vehicle obtains, from the CA, a master key and a master certificate on that key. After that, the vehicle uses its master key, its master certificate and the public key of the CA in order to generate any pseudonym and the CA’s corresponding certificate on that pseudonym. No communication with the CA is necessary for that purpose. The authors refer to this modified PKI as PKI+. Autonomous pseudonym creation has been made possible by using an elliptic-curve-based scheme from Zeng (2006). The authors mention that about $2^{170}$ certified pseudonyms can be generated autonomously by any vehicle. PKI+ also makes certificate revocation easier. No CRLs are now maintained. Instead an amount of data slightly more than one kilobyte needs to be transmitted to each vehicle in order to permanently revoke a node (along with all its certificates) from the network.
6.3 Non-repudiation

While the problems of privacy and unlinkability have been satisfactorily solved by using pseudonyms and reanonymisation, the question of traceability continues to remain unaddressed. It is possible to incorporate this requirement in the framework of pseudonymous authentication. The idea is to store some information in each certificate that binds a vehicle’s identity to the certificate in a non-obvious manner. This binding cannot be understood by other vehicles nor by transport agencies. However, higher authorities (like the court) may use this information to identify a vehicle.

A suggestion for a concrete realisation of this invisible identity comes from Rahman and Hengartner (2007). They encrypt the vehicle’s identity by secret keys of the court and the government transport authority. The encrypted identity is attached to each certificate given to the vehicle. The court and the transport authority must cooperate in order to retrieve the vehicle’s identity. In order that the invisible identity cannot be used to track a vehicle, a randomised encryption scheme (Rivest and Sherman, 1983) is to be used. Such a scheme produces different ciphertexts during different trials even when the plain text and the key remain the same.

7 Some special forms of authentication

Other authentication models often help a VANET in several ways. In this section, we describe two such variants.

7.1 Group communication

So far, we have discussed schemes where each vehicle sends authenticated messages along with its own (temporary) identity. It is also possible that vehicles form groups and communicate on behalf of the groups. Messages are now authenticated by the identity of the group. The identity of individual vehicles is not sent, that is privacy is enhanced. Moreover, members in a group may share symmetric keys. Symmetric encryption among group members allows secure and efficient communication. The articles (Raya et al., 2006a; Calandriello et al., 2007; PlöBl and Federrath, 2008) address the issues of group communication in a VANET.

7.1.1 Group formation

Vehicle groups may be static or dynamic. Static groups may correspond to vehicle manufacturers, transport authorities or types of vehicles.

Dynamic groups, on the other hand, may be based upon locations. Raya et al. (2006a) proposed an on-the-fly group formation based upon fixed geographic regions. Vehicles situated in a region form the current group associated with the region. Since vehicles are assumed to be equipped with location-finding devices like GPS, each vehicle knows the group(s) to which it belongs. In order to maintain overall connectivity, two adjacent groups need to be overlapping. The leader of a group is selected to be the vehicle closest to the centre of the group region (assumed circular). In case of ties or close ties, some other arbitrating parameters (like vehicle identities) need to be taken into account.
The group leader obtains a certified public-key pair from the CA and distributes the key pair to other members of the group. This transmission should be encrypted by the public keys of the group members and signed by the private key of the leader. Once a group key is established, group members may communicate using symmetric techniques, like HMACs.

7.1.2 Group signatures
Calandriello et al. (2007) proposed two authentication schemes involving group identities. In their first scheme, a vehicle signs a message by the public key of the group to which it belongs. In the second scheme (called the hybrid scheme), a vehicle generates a pseudonym, authenticates the pseudonym by the group public key and signs the message with the pseudonym.

7.2 Data aggregation
If vehicles transmit individual messages, each vehicle needs to authenticate and analyse all the messages received from all of its neighbours. This is called destination aggregation. Raya et al. (2006a) proposed replacing this technique by source aggregation. Suppose that a vehicle $U_1$ sends a message $M$ (say, about traffic condition). Upon receiving (an authenticated copy of) $M$, a second vehicle $U_2$ realises that it too needs to send the same message. So $U_2$ forwards the message to its neighbours. When a third vehicle $U_3$ receives the forwarded message (possibly also the original message from $U_1$), it re-forwards the message to its respective neighbours, and so on. The basic idea is the replacement of new messages by forwarded messages. Simulation studies conducted by Raya et al. (2006a) and Eichler et al. (2006) indicated that this practice reduces the communication overhead for the transmission of $M$.

Every forwarded message must support the authenticity of each vehicle that takes part in the creation of the message. In other words, the certificates of all the vehicles $U_1$, $U_2$, $U_3$, … mentioned above must be present in the forwarded message. A concatenated signature on $M$ looks like

$$(S_1, C_1, S_2, C_2, S_3, C_3, \ldots),$$

where $S_i$ is the signature of $U_i$ on $M$ and $C_i$ is the certificate of $U_i$. An onion signature, on the other hand, is generated as follows. The vehicle $U_1$ generates its original signature $S_1$ on $M$ and sends $(M, S_1, C_1)$, whereas $U_2$ signs $S_1$ and sends $(M, S_2, S_1, C_1, C_2)$, $U_3$ signs $S_2$ and sends $(M, S_3, S_2, C_1, C_2, C_3)$ and so on. More generally, the $i$-th vehicle $U_i$ in the process generates the signature as follows:

$$S_i = sgn_i(S_{i-1})$$

$$= sgn_i\left(sgn_{i-1}\left(sgn_{i-2}\left(\cdots\left(sgn_1(M)\right)\cdots\right)\right)\right)$$

In order that $S_i$ can be verified, $U_i$ needs to send the previous signature $S_{i-1}$. The signatures $S_{i-2}$, $S_{i-3}$, ..., $S_1$ are not sent by $U_i$. That is, irrespective of the length of the forwarding chain, the authenticated message always contains only two signatures. This makes onion signatures more compact than concatenated signatures. On the other hand,
the authenticity of each $U_i$ can be verified independently in a concatenated signature. In case of an onion signature, an invalid signature made by $U_i$ invalidates all subsequent signatures $S_{i+1}, S_{i+2}, \ldots$.

Data aggregation can also be used for other security purposes like detection of malicious nodes and bogus information (e.g. Eichler et al., 2006; Picconi et al., 2006).

8 Conclusion

In this paper, a survey of several known techniques for secure authentication of messages in a VANET is presented. The approach based on public-key signatures meets all the relevant security objectives in a reasonable manner and happens to be the technique that is most widely studied and recommended by the research community. However, there remain significant scopes for tuning public-key-based authentication so as to make it more suitable for deployment in VANETs. We also describe the use of hash chains and hash trees for authentication in VANETs. To the best of our knowledge, these alternative techniques lack comprehensive treatment in the current literature.

We concluded this paper by highlighting some promising directions of future research:

1. The basic disadvantage of digital signatures is that they incur significant computation overhead during signature generation and verification. Improvements, possibly at the cost of an acceptable degradation of security, are solicited. Note that most messages sent by vehicles do not require very long-term security.

2. The issues of distribution and storage of CRLs need to be addressed further.

3. The provision of revocation of certificates needs to be embedded in authentication schemes based on hash trees. More concretely, a complete security architecture addressing all the security requirements in VANETs needs to be developed for hash trees.

4. Other alternative authentication models may be investigated.

5. Although group communication promotes privacy and unlinkability, the issues pertaining to traceability are ignored. Possibilities of efficiently embedding sender-specific information in group signatures may be investigated.

References


Authentication schemes for VANETs


Authentication schemes for VANETs


