Cryptanalysis and Efficient Improvement of a Robust and Scalable One-Way Hash Chain Authentication Protocol in Vehicular Communication

Lo-Yao Yeh\textsuperscript{1}, Woei-Jiunn Tsaur\textsuperscript{2}, and Tong-Ying Juang\textsuperscript{3}

\textsuperscript{1}Network and Security Division, National Center for High-Performance Computing (NCHC), Taichung, Taiwan
\textsuperscript{2}Department of Information Management, Da-Yeh University, Changhua, Taiwan
\textsuperscript{3}Department of Computer Science and Information Engineering, National Taipei University, Taipei, Taiwan

Abstract—Recently, a hash-chain based authentication protocol has been proposed, named HAP, in vehicular communications to improve the scalability. By means of removing certificate cost, HAP earns a great performance in terms of computational overhead and message loss ratio. Unfortunately, we discover a fatal security flaw that malicious vehicles can masquerade as an innocent one to disseminate forged messages, which contradict to the security claims of HAP. This study tries to use the least effort to mend the severe security leak. We deliberately strengthen the security level of HAP by only adding a random number in registration phase without any harm to its excellent performance result.

Keywords: VANETs, Authentication, Hash-chain, Key Reveal Attack, Batch Verification, Scalability.

1. Introduction

With the rapid development of wireless communication, Vehicular Ad Hoc Networks (VANETs) has gripped attention from scholars, industries, and governments. Each vehicle with On Board Unit (OBU) could serve as a wireless router, node, or sensor to share road information with other vehicles and roadside units (RSUs). The trusted authority (TA), such as ministry of communications, can gather real-time road information and instruct alternative routes to other vehicles while a traffic accident occurs. Therefore, the message validity is a significantly important issue for VANETs. In general, digital signature technique \cite{1}, \cite{2}, \cite{3}, \cite{4}, \cite{5} is wildly adopted to verify the validity of the received messages owing to its merits of authentication and non-repudiation. With a good digital signature scheme, a receiver is able to check the correctness of the gained message and the legitimacy of the signer. No adversary can masquerade as other legal signer to forge a signature on a message without knowledge of the private key. On the other hand, according to DSRC \cite{6}, a vehicle sends a message within the time interval of 100-300 ms. A vehicle or RSU may receive a huge quantity of messages in a short period. Hence, the verification time of a digital signature scheme for VANETs is another considerable topic.

In 2013, a hash-chain based authentication protocol, named (HAP), was proposed to greatly reduce the computational overhead. The batch verification \cite{7} is also supported for fast signature verification. Moreover, the packet size of each message can be reduced by 50.1\% due to getting rid of the attached certificate. In HAP, RSU is responsible for generating a series of public/private key pairs based on a one-way hash chain scheme and distributing them along with a \(n\) bit hash code \(H\) and a proof cipher \(\hat{C}\) to the vehicles within its range. Vehicles can verify the received message and its sending vehicle by the attached public key and its hash code \(\hat{H}\).

The study discloses a fatal security problem called private key reveal attack in HAP. In order to omit the certificate cost in packet size, HAP adopts a one-way hash chain to produce the private key \(Pr\) and corresponding public key \(Pu\). Each hash code \(\hat{H}\) is used to ensure the ownership of its corresponding public key \(Pu\). The other vehicle keeps the same proof cipher \(\hat{C}\) to confirm the relationship between \(\hat{H}\) and \(Pu\). However, owing to using a traditional one-way hash chain, an adversary within the same RSU range can exploit the one-way hash function property to guess the following possible private keys \(Prs\) and take advantage of the proof cipher \(\hat{C}\) to affirm the guessing result. Therefore, the all claimed security features, including source authentication, message integrity, and movement tracking etc., in HAP are deemed to be insecure. To overcome this serious security problem, we propose an improved key generation method to enhance the security of HAP. It is worth to mention that our improved scheme is with “no extra verification time” for vehicles and RSUs in the process of execution. Note that we believe the best improvement scheme is with the least effort to achieve a significant security leap. We also provide a security analysis to evaluate the strength of the proposed scheme.

The remainder of this paper is organized as follows. Section 2 briefly introduces the hash-chain based authentication protocol (HAP). In Section 3, we demonstrate the HAP is vulnerable to the private key reveal attack. In Section 4, we present an improved authentication scheme, and the security and performance analysis are shown in Section 5. Finally, a
conclusion is marked in Section 6.

2. Review of hash-chain based authentication (HAP) protocol

According to IEEE 1609.2 standard [2] for VANETs, every message comprises of a 2 bytes header, a 67 bytes payload/message, a 56 bytes digital signature, and a 126 bytes certificate, which means the overhead of certificate occupies a half of the total overhead. Recently, a traditional hash-chain based authentication (HAP) [8] protocol was proposed to significantly mitigate the burden of certificate.

2.1 Vehicle and RSU Registration

For a vehicle $V_x$, the trusted authority (TA) will generate its public key $PK_{V_x} = g^{sk_{V_x}}$, corresponding private key $sk_{V_x} \in Z^*_p$, and certificate $Cert_{TA}[PK_{V_x}]$. As for RSU $R$, TA also issues the public key $PK_R = g^{sk_R}$, corresponding private key $sk_R \in Z^*_p$, and certificate $Cert_{TA}[PK_R]$. Note that each RSU maintains an adjacent RSU list with the IDs of all its neighboring RSUs, and their corresponding proof ciphers $\hat{C}$ for facilitating the inter-RSU vehicle communications.

2.2 Key Management using One-way Hash Chains

The RSU $R$ chooses a secret integer $s \in \{0, 1, ..., p - 1\}$ and computes a one-way hash function on the secret $s$ for a large $k$ as below. $h^i(s)$ means that the one-way hash function $h$ is applied on $s$ for $i$ number of times.

$$h^i(s) \leftarrow h^{i-1}(s) \cdot ... \cdot h^1(s) \leftarrow h(s) \leftarrow s$$

The $R$ then computes a total of $k - 1$ anonymous private/public key pairs and their corresponding hash codes $\hat{H}$ from the generated one-way hash chain. The $j$th anonymous private key $Pr_j = \prod_{i=0}^j h^i(s)$ and its corresponding public key $Pu_j = g^{Pr_j} = g^{\prod_{i=0}^j h^i(s)}$ . The $k - 1$ anonymous private/public key pairs and their corresponding $n$ bit hash codes $\hat{H}_j = \prod_{i=j+1}^k h^i(s)$ are also computed, and $\hat{C} = F(g^{\prod_{i=0}^k h^i(s)})$, where $F()$ is a cryptographic one-way hash function such that SHA-1. For example,

<table>
<thead>
<tr>
<th>$j = 1$</th>
<th>$j = 2$</th>
<th>$j = k - j - 1$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$Pr_1 = \prod_{i=0}^1 h^i(s)$</td>
<td>$Pr_2 = \prod_{i=0}^2 h^i(s)$,</td>
<td>$Pr_{k-j-1} = \prod_{i=0}^{k-j-1} h^i(s)$,</td>
</tr>
<tr>
<td>$Pu_1 = g^{\prod_{i=0}^1 h^i(s)}$</td>
<td>$Pu_2 = g^{\prod_{i=0}^2 h^i(s)}$,</td>
<td>$Pu_{k-j-1} = g^{\prod_{i=0}^{k-j-1} h^i(s)}$,</td>
</tr>
<tr>
<td>$\hat{H}<em>1 = \prod</em>{i=j+1}^k h^i(s)$</td>
<td>$\hat{H}<em>2 = \prod</em>{i=j+1}^k h^i(s)$,</td>
<td>$\hat{H}<em>{k-j-1} = \prod</em>{i=j+k-j}^k h^i(s)$</td>
</tr>
</tbody>
</table>

2.3 Vehicle-to-Infrastructure (V2I) Communication

1) $R \rightarrow V_x$: Hello message$|Cert_{TA}[PK_R]$.

Entering into $R$'s communication range, $V_x$ can receive the periodic hello message, including the public key, location, timestamp etc., of $R$. Then, $V_x$ starts the mutual authentication protocol and generates a shared session key $SS_{VR}$ based on Diffie-Hellman key establishment protocol [9]. $V_x$ first sends the encrypted message $E_{PK_R}(PK_{V_x}||Cert_{TA}[PK_{V_x}]||g^x||m) = 1$ (including the share parameter for the session key, $x$).

2) $V_x \rightarrow V_y$: $E_{PK_{V_x}}(PK_{V_y}||Cert_{TA}[PK_{V_y}]||g^x||m) = 2$.

After decrypting the message and checking the validation of $V_x$’s public key and signature, $V_y$ chooses $g^y$ and $m$ sets of private/public key pairs, their corresponding hash codes $\hat{H}$ from the list of $k - 1$ key sets it generated, as well as the final proof cipher $\hat{C}$. Then, $R$ encrypts them with the $PK_{V_x}$ and sends then encrypted message $E_{PK_{V_x}}(g^x||g^y||\hat{H}_1||\hat{C})||\hat{m}_1 = 3$ (including $\hat{m}$) to $V_x$.

3) $V_x \rightarrow V_y$: $E_{PK_{V_x}}(g^y||\hat{H}_1||\hat{C})||\hat{m}_1 = 4$.

$V_x$ decrypts the message and confirms the correctness of the signature $\sigma_{sk_R}$. If the above three steps are completed correctly, the mutual authentication succeeds.

Note that each RSU maintains an adjacent RSU list with the IDs of all its neighboring RSUs, and their corresponding proof ciphers $\hat{C}$ for facilitating the inter-RSU vehicle communications.

2.4 Vehicle-to-Vehicle (V2V) Communication

In V2V communication, $V_x$ picks a public/private key $Pu_i/Pri_i$ pair and generates the message $msg||Pu_i||\hat{H}_1||RSU_{ID}||\hat{m}_1||V_y$ to $V_y$. If the $V_y$’s RSU$_{ID}$ is the same as that of $V_x$, $V_y$ can directly authenticate $V_x$ by checking the validity both $\sigma_{Pri_i}$ and $\hat{C}_x$. Note that the proof cipher $\hat{C}_x$ is computed from $Pu_i$ and $\hat{H}_1$. As long as the computed $\hat{C}_x$ matches its own $\hat{C}_y$, $V_y$ will trust $V_x$ and consume the message. If the $V_x$’s RSU$_{ID}$ is different from that of $V_x$, $V_y$ then forwards the received message to its RSU for the message authentication. Next, the RSU computes the hash of proof cipher $\hat{C}_x$ from the forwarded message and matches it with its adjacent RSU’s corresponding proof cipher. If only those proof ciphers match, the RSU delivers a reply message to the requesting vehicle to confirm the authenticity of the message.

2.5 Batch Verification

In HAP, the batch verification is provided and each receiving vehicle needs to perform two operations including (1) batch signature verification, and (2) batch proof ciphers verification to complete the message authentication. If a vehicle wants to send a message $M$ with the public key $Pu_i$, the OBU computes $\omega = g^b$, where $b \in Z^*_p$, and $\alpha = Pr_i + b\hat{\beta}$, where $\beta = h(M||Pu_i||\alpha||TS)$ and $TS$ is the timestamp. Then, $\sigma_{Pri_i}(\omega, \alpha)$ is a valid signature on $M$. 

Any receiving vehicle can verify the signature if
\[ \hat{e}(g, \omega)\beta = \hat{e}(\omega, Pu_1 \cdot \omega^\beta) \]
as verified below:
\[
\begin{align*}
\hat{e}(g, \omega)^\alpha &= \hat{e}(g, \omega)^{(Pr_{r_1}+\beta\beta)} = \hat{e}(g, \omega)^{Pr_{r_1} \cdot \hat{e}(g, \omega)^{\beta}} \\
&= \hat{e}(g^Pr_{r_1}, \omega) \cdot \hat{e}(g, \omega^{\beta}) = \hat{e}(\omega, Pu_1) \cdot \hat{e}(\omega, \omega^\beta) \\
&= \hat{e}(\omega, Pu_1 \cdot \omega^\beta)
\end{align*}
\]
Consider the receiver receives \((\omega_1, \alpha_1), (\omega_2, \alpha_2), ..., (\omega_d, \alpha_d)\) on the messages \(M_1, M_2, ..., M_d\), respectively. To perform the batch signature verification, the receiving vehicle calculates \(\omega' = \prod_{i=1}^{d} \beta_i \omega_i, \alpha' = \sum_{i=1}^{d} \alpha_i\), and checks whether
\[ \hat{e}(g, \omega')^{\alpha'} = \hat{e}(\omega', \prod_{i=1}^{d} Pu_i \cdot \omega_i^\beta). \]

For single and batch proof ciphers verification, given a public key \(Pu_2\) and its corresponding hash code is \(\tilde{H}_x\) (Assume \(x = k - j\)), the receiver can verify the integrity of the public key by \(\tilde{C}_x = h(Pu_2)\tilde{H}_x = h(Pu_{k-j})\tilde{H}_{k-j} \equiv h(g^{\sum_{i=1}^{k-j} h(s)^i}) = F(\sum_{i=1}^{d} Pu_i) = \hat{C}_y\).

The details of formula verification can be found in [8].

3. Security Weakness of HAP

In HAP [8], the authors claim several security properties such as source authentication, message integrity, and resistance to replay attack. However, this corresponding paper demonstrates an attack to breach all claimed security features because of the leakage of the private key.

Assume that a malicious vehicle \(V_m\) obtains \(m\) sets of private/public key pairs and corresponding hash code \(\tilde{H}\) from RSU. The following procedures can show how to derive the private keys of some other vehicles.

Step1. \(V_m\) takes \(Pr_{r_0} = \prod_{i=0}^{k-1} h^i(s)\) and \(Pr_{r_{a-1}} = \prod_{i=0}^{a-1} h^i(s)\), and divides them to attain \(h^a(s)\).

Step2. Then, \(V_m\) inputs the hash value \(h^a(s)\), \(\tilde{H}_a\), and the final proof cipher \(\tilde{C}\) into Algorithm 1 to derive the compromised private key \(Pr_c\) and corresponding hash code \(\tilde{H}_c\). Now, it is easy to generate the public key \(Pu_c = g^{Pr_c}\).

For example:

Assume that a malicious vehicle \(V_m\) obtains two private/public key pairs, \(Pr_{r_19}\) and \(Pr_{r_20}\), corresponding hash codes \(\tilde{H}_{19}\) and \(\tilde{H}_{20}\), and the proof cipher \(\tilde{C} = F(g^{\sum_{i=b}^{a} h^i(s)})\) from RSU.

Algorithm 1 Derive algorithm
- Input: hash value \(h^s(s)\), \(\tilde{H}_a\), and the final proof cipher \(\tilde{C}\).
- Output: the compromised \(Pr_c\) and corresponding \(\tilde{H}_c\).

1 Derive Alg(m):
2 For \(x = 1\) to \(Z\) (where \(Z\) is a large number)
3 performs \(x\) times hash function on \(h^s(s)\) to get \(h^{a+x}(s)\);
4 computes \(Pr_{r_{a+x}} = Pr_{r_{a+x-1}} \cdot h^{a+x}(s)\) and computes \(\tilde{H}_{a+x} = \tilde{H}_{a+x-1}/h^{a+x}(s)\);
5 If \((\tilde{C} = F(g^{Pr_{r_{a+x}}}) = F(\tilde{H}_{a+x}))\) returns the compromised key pair <\(Pr_{r_{a+x}}, \tilde{H}_{a+x}\)> else end;
6 Next \(x\).

Step1. \(V_m\) takes \(Pr_{r_20} = \prod_{i=0}^{20} h^i(s)\) and \(Pr_{r_{19}} = \prod_{i=0}^{19} h^i(s)\), and then \(h^{20}(s) = Pr_{r_{20}}/Pr_{r_{19}} = \prod_{i=0}^{19} h^i(s)\) and \(\tilde{H}_{20} = \tilde{H}_{20}/h^{21}(s) = \prod_{i=21}^{20} h^i(s)\).

Step2. \(V_m\) performs one time hash function on \(h^{20}(s)\) to get \(h^{21}(s) = h(h^{20}(s))\).

Step3. \(V_m\) computes \(Pr_{r_{21}} = Pr_{r_{20}} \cdot h^{21}(s) = \prod_{i=0}^{20} h^i(s) \cdot h^{21}(s)\), and calculates \(\tilde{H}_{21} = \tilde{H}_{20}/h^{21}(s) = \prod_{i=21}^{20} h^i(s)\).

Step4. \(V_m\) checks if \(F(g^{Pr_{r_{21}}} \tilde{H}_{21}) = F(g^{\prod_{i=0}^{19} h^i(s) \cdot h^{21}(s)}) = F(g^{\sum_{i=a}^{b} h^i(s)}) = \tilde{C}\) holds, the compromised key pair <\(Pr_{r_{21}}, \tilde{H}_{21}\)> are earned.

As a result, \(V_m\) can disguise himself as the other vehicle to sign false messages by <\(Pu_{a_c}, Pr_{r_c}, \tilde{H}_c\)>, which means the whole security foundation of HAP has been broken. Note that the sequence of \(m\) sets of private/public key pairs is never discussed in HAP so it can reasonably infer those are successive. Even if those key pairs are randomly picked, the above attack still works. Using the hash function properties, attackers just need more time to determine the sequence of any two key pairs.

4. The Improved Scheme

In this short paper, we adhere to keep the merits of efficiency in HAP scheme [8] in the sense that we come up with a simple but important amendment to fix above security flaw. Instead of complex reformation, our improved scheme increases a lightweight operation in key management phase. The key to performing the proposed private key reveal attack is the “inferable hash chain.”

Therefore, we devise that each RSU adopts a non-inferable hash chain to withstand the attack. In key management phase, RSU can pick a random number \(r \in \mathbb{Z}_p^*\) to compute a key-based one-way hash chain on secret \(s\) as follows.
In the improved scheme, each RSU in the key management phase only needs to pick a new random number \( r \in \mathbb{Z}_p^* \) to form a key-based one-way hash chain. The rest parts of HAP stay the same in the sense that the additional performance cost is almost negligible. The summary of improvement cost is listed in Table 1.

<table>
<thead>
<tr>
<th>Phases</th>
<th>Cost for improved scheme</th>
</tr>
</thead>
<tbody>
<tr>
<td>Vehicle and RSU registration phase</td>
<td>1 random number (negligible)</td>
</tr>
<tr>
<td>V2I communication phase</td>
<td>None</td>
</tr>
<tr>
<td>V2V communication phase</td>
<td>None</td>
</tr>
<tr>
<td>Batch verification phase</td>
<td>None</td>
</tr>
</tbody>
</table>

### 5.1 Resistance to the Proposed Private Key Reveal Attack

In the proposed attack, HAP will reveal the private key to other vehicles within the same RSU range, which incurs the failure of source authentication, message integrity, movement tracking prevention and traceability. In order to overcome this fatal weakness, we design a new generation method of one-way hash chain while RSU produces a hash chain value by this formula \( h^{k}(s) \leftarrow h(h^{k-1}(s), r), \) where \( r \in \mathbb{Z}_p^* \). In that case, an attacker cannot reason the possible private keys and hash chain values \( \tilde{H} \). Hence, the proposed improved scheme can withstand the proposed private key reveal attack.

### 5.2 Source Authentication and Message Integrity

In the improved scheme, the source of message can be confirmed by (1) the correctness of the signed signature, and (2) the integrity of public and private key. First, the receiver will check the signature \( (\omega, \alpha) \) by the formula \( \hat{e}(q, \omega) = \hat{e}(\omega, Pu_c \cdot \omega^j) \). If the signature is valid, it means that the sender possesses the corresponding private key \( Pr_c \). Next, to avoid the attachment of a separate public key certificate, the receiver also examines the relationship of \( \tilde{H}_c \) and \( Pu_c \) by the formula \( \tilde{C}_c = h(Pu_c \cdot \tilde{H}_c) = \tilde{C}_r \). In terms of message integrity, the public key is used to encrypt messages delivered by vehicles and RSUs. As long as the security of private key is provided, the message integrity is also guaranteed.

### 5.3 Performance Analysis

In the improved scheme, each RSU in the key management phase only needs to pick a new random number \( r \in \mathbb{Z}_p^* \) to form a key-based one-way hash chain. The rest parts of HAP stay the same in the sense that the additional performance cost is almost negligible. The summary of improvement cost is listed in Table 1.