Accountable and Cross-checking Long Term Certification Authority (Interim Version)

Keyao Huang
KTH Royal Institute of Technology
Stockholm, Sweden
Keyao@kth.se

***Abstract*—**

I. Introduction

Vehicular Communication (VC) systems are intended to facilitate the exchange of information between traffic entities, avoid potential road dangers, and improve safety when vehicles are in motion. Common models of communication technologies include vehicle-to-vehicle (V2V) and vehicle-to-infrastructure (V2I). The Vehicle Public Key Infrastructure (VPKI) provides certificates to entities in VC systems, primarily through public-private key pairs, to ensure the confidentiality, integrity, and authenticity of communications, and to protect against unauthorized data transmission and data tampering. There are many designs and schemes for VPKI, this report focuses on the SECMACE architecture [1] and intends to address the shortcomings therein. SECMACE considers multi-domains and Each domain consists of a Long Term Certification Authority (LTCA), several Pseudonym Certification Authority (PCA), and a Resolution Authority (RA). Briefly, the workflow is as follows: (1) Vehicle registers with LTCA to obtain a long-term certificate (LTC). (2) The vehicle obtains a ticket from LTCA. (3) The vehicle obtains pseudonyms from PCA by leveraging the ticket. Besides, RA can interact with CAs for pseudonym resolution and revoke the process when it receives a report of a suspicious pseudonym.

Issues and Challenges: In SECMACE [1] and SECMACE+ [2], although the RA can resolve and revoke an illegal pseudonym, this process is triggered only when entities report that a suspicious pseudonym is detected, such as unusual data in the beacon. That means if the illegal pseudonym is not perceived in the VC communication, RA would not perceive CAs' misbehavior and accuse them. In that case, a compromised CA can still issue malicious pseudonyms or tickets normally until the illegal pseudonym is perceived in the VC system. Illegal data that is signed with an illegal pseudonym has been transmitted in the VC communication until it is detected, which would pollute the VC environment.

II. Adversary Model & Problem Definition

Adversary Model: This report considers malicious behaviors that a compromised LTCA can perform. Assuming the PCA is trusted, it will strictly adhere to the pseudonym release protocol. A compromised LTCA could (1) register an illegal vehicle and issue an LTC or ticket that will be used to acquire pseudonyms from PCA, (2) issue multiple simultaneously valid tickets for one vehicle. the PCA would not be able to identify that multiple tickets are bound to the same LTC upon receipt of the tickets, so the PCA would issue pseudonyms for multiple concurrently valid tickets. (3) Return different LTCs during the RA resolution process, thus misleading the system.

Problem Definition: In SECMACE [1] and SECMACE+ [2], the LTCA checks if a valid ticket has been issued for the LTC and prevents the malicious vehicle (external adversary) from obtaining a concurrently valid ticket, which is based on the assumption that the LTCA is honest. However, if LTCA is compromised (internal adversary), it can deviate from the protocol or policy, and issue simultaneously valid tickets for the malicious vehicle. Therefore, we have to consider the scenario where LTCA is compromised. And prevent the malicious behavior of LTCA and hold it accountable for the malicious behavior to improve the security of VPKI.

III. Related Work

Accountable Key Infrastructure: In the AKI proposal [3], the Log Server uses integrity trees to log certificates issued by CAs and introduces validators to monitor the Log Server and check it for malicious behavior (e.g. not logging certificates). However, it does not consider the case where the Log server and validators are compromised at the same time. ARPKI [4] extends AKI and introduces multiple Log Servers to record certificates. Synchronizing certificate information through multiple Log Servers ensures the singularity of certificates, prevents mimicry attacks, and maintains the accountability of the certificate. The authors use CA1 as a bridge between certificate requester and Log Server, CA1's main role is to forward the certificate acquisition request to the Log Server, which is responsible for logging the request and synchronising it with other Log Servers. After most Log Servers complete the registration, the confirmation of acceptance will be sent to CA2, which will sign the certificate and return it to CA1 for signing again. The final certificate format is $((Accept)_{\sigma CA2})_{\sigma CA1}$. CA1 and CA2 do not generate certificates, but only sign the acceptance acknowledgements generated by Log Server. **Therefore, the main idea of ARPKI is to use two CAs to check against the Log Server to confirm that the Log Server has accepted and logged the user's certificate request. This approach does not ensure that the CAs are behaving correctly**, as the CAs

do not generate certificates, but rather sign the Log Server's acknowledgement of receipt. This scheme cannot be directly applied to VPKI because in VPKI the vehicle certificates are generated by the CA. And cross-checking in ARPKI does not check the CA's misbehaviour. My proposed scheme draws on the ideas it proposed to record the behavior of LTCA and cross-check the LTC or ticket before issuing.

VPKI: ACMS [5] is a VPKI scheme that uses a transparent Log Server to ensure certificate transparency. It uses multiple LTCAs to prevent the issue of a single LTCA being compromised. The vehicle sends the registration request to LTCAs and obtains signatures from multiple LTCAs. Then submits them to the validator and obtains a long-term identity after verification. However, the authors do not discuss the issue of the validator being compromised, where a malicious validator can forward the registration request with multiple LTCA signatures that are less than the threshold to the Log server and allow the vehicle to complete the registration. IOTA-VPKI [6] applies distributed ledger technology with SECMACE scheme [1] to increase the transparency of CA-signed certificates. It records $S = (Sign(LK_{CA}, H(crt)), Id_{CA})$ on the distributed ledger, and it is not able to check the information inside the crt because only H(crt) is recorded. Moreover, there is no checking mechanism to verify the behavior of LTCA in generating LTC and tickets, so LTCA's malicious behavior cannot be detected. Finally, the scheme is not experimentally verified and security analyzed. SCMS [7] focuses on V2X communication security and proposes a novel approach called butterfly key expansion that allows on-board equipment (OBE) to only send a single request to obtain multiple pseudonyms, which avoids the need for the vehicle to generate thousands of public keys and send them to the VPKI. The authors consider the malicious behavior of vehicles and propose a targeted detection mechanism, but the authors do not consider the malicious behavior of LTCA.

IV. Proposed Scheme Overview

A. Preliminaries & Assumptions

This project mainly focuses on one domain. The premise of this scheme is that there are two LTCAs in each domain, both of which can register vehicles and issue tickets. A Log Server is introduced to record the behavior of LTCAs and the issued certificates or tickets, called LTC-Tickets Log Server (LTLS). The architecture of this scheme is shown in Fig. 1. Data communications between entities in the system (LTCAs and LTLS) are transmitted through end-to-end authenticated Transport Layer Security (TLS) [8] channels, which ensures data confidentiality and integrity. The notations that appear later in the text are listed in table **??**.

B. Workflow

Vehicle sends a request to $LTCA_{Gen}$ ($LTCA_1$ in Fig. 1) to register/get tickets (Step 1). LTCA generates LTC / ticket according to Protocol 1 / Protocol 3 in SECMACE+ [2], then sends the output data including its generating time, $LTCA_Id$, which makes $LTCA_{Gen}$'s behavior accountable, to $LTLS$ (Step 2). Upon receiving the message, $LTLS$ caches it and checks the behavior of the $LTCA_{Gen}$ in generating the TLC / ticket and signs the TLC / ticket. After signing, $LTLS$ sends the same message to $LTCA_{Check}$ ($LTCA_2$ in Fig. 1) as well as its signature (Step 3). $LTCA_{Check}$ verifies $LTLS$'s signature and performs the same checking operation as $LTLS$ does to verify the correctness of $LTCA_{Gen}$'s generating behavior. After that, $LTCA_{Check}$ signs the LTC / ticket and sends the message with its signature to $LTCA_{Gen}$ and $LTLS$ concurrently (Step 4 & Step 5). $LTCA_{Gen}$ sends the LTC / ticket to the vehicle with three signatures ($LTCA_{Gen}$, $LTCA_{Check}$ and $LTLS$) (Step 6). Upon receiving the message from $LTCA_{Check}$, $LTLS$ logs it including three signatures. In this case, the three entities form a chain of accountability, and they are jointly responsible for the issuance of the upcoming LTC / ticket. If in the future a malicious behavior of the vehicle causes the RA to resolve the LTC / ticket and it is found to have been issued by the $LTCA_{Gen}$ through a malicious behavior, then all three entities will be held accountable and punished together.

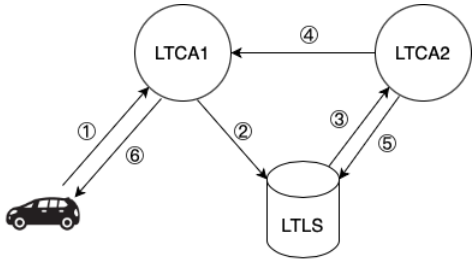

Fig. 1. Architecture

C. Storage Scheme

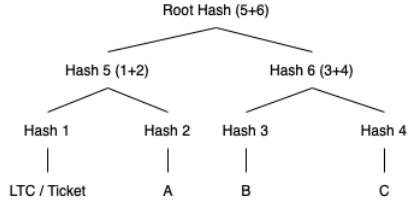

Fig. 2. MHT

V. Expected Results

The compromised LTCA cannot generate and issue LTC and tickets for illegal vehicles through the mutual checking between the two LTCAs and the $LTLS$. The double checking of the $LTCA_{Gen}$'s behaviour by $LTCA_{Check}$ and $LTLS$ can effectively prevent the malicious behaviour of $LTCA_{Gen}$. Secondly $LTLS$ compares the LTC/ticket generated by $LTCA_{Gen}$ with the data recorded in the database, which can detect whether $LTCA_{Gen}$ is trying to generate concurrently valid tickets. Thirdly, RA can compare the obtained data with the hash value in the Merkle Hash Tree during resolution. This prevents $LTLS$ from returning incorrect data that could mislead the RA.

VI. Qualitative Analysis

A. Overheads

VII. Conclusion

References

[1] M. Khodaei, H. Jin, and P. Papadimitratos, "Secmace: Scalable and robust identity and credential management infrastructure in vehicular communication systems," *IEEE Transactions on Intelligent Transportation Systems*, vol. 19, no. 5, pp. 1430–1444, 2018.

[2] M. Khodaei, H. Noroozi, and P. Papadimitratos, "Secmace+: Upscaling pseudonymous authentication for large mobile systems," *IEEE Transactions on Cloud Computing*, 2023.

[3] T. H.-J. Kim, L.-S. Huang, A. Perrig, C. Jackson, and V. Gligor, "Accountable key infrastructure (aki) a proposal for a public-key validation infrastructure," in *Proceedings of the 22nd international conference on World Wide Web*, pp. 679–690, 2013.

[4] D. Basin, C. Cremers, T. H.-J. Kim, A. Perrig, R. Sasse, and P. Szalachowski, "Design, analysis, and implementation of arpki: an attack-resilient public-key infrastructure," *IEEE Transactions on Dependable and Secure Computing*, vol. 15, no. 3, pp. 393–408, 2016.

[5] S. Khan, L. Zhu, X. Yu, Z. Zhang, M. A. Rahim, M. Khan, X. Du, and M. Guizani, "Accountable credential management system for vehicular communication," *Vehicular Communications*, vol. 25, p. 100279, 2020.

[6] A. Tesei, L. Di Mauro, M. Falcitelli, S. Noto, and P. Pagano, "Iota-vpki: A dlt-based and resource efficient vehicular public key infrastructure," in *2018 IEEE 88th Vehicular Technology Conference (VTC-Fall)*, pp. 1–6, IEEE, 2018.

[7] B. Brecht, D. Therriault, A. Weimerskirch, W. Whyte, V. Kumar, T. Hehn, and R. Goudy, "A security credential management system for v2x communications," *IEEE Transactions on Intelligent Transportation Systems*, vol. 19, no. 12, pp. 3850–3871, 2018.

[8] E. Rescorla, "The transport layer security (tls) protocol version 1.3," tech. rep., 2018.