SECMACE+: Upscaling Pseudonymous Authentication for Large Mobile Systems

Mohammad Khodaei, *Member, IEEE,* Hamid Noroozi, and Panos Papadimitratos, *Fellow, IEEE*

Abstract—The central building block of secure and privacy-preserving Vehicular Communication (VC) systems is a Vehicular Public Key Infrastructure (VPKI), which provides vehicles with multiple anonymized credentials, termed *pseudonyms*. These pseudonyms are used to ensure VC message authenticity and integrity while preserving vehicle (thus passenger) privacy. In the light of emerging large-scale multi-domain VC environments, the efficiency of the VPKI and, more broadly, its scalability are paramount. By the same token, preventing misuse of the credentials, in particular, Sybil-based misbehavior, and managing *"honest-but-curious"* VPKI entities are other facets of a challenging problem. In this paper, we leverage the state-of-the-art VPKI system and *enhance* its functionality towards a highly-available, dynamically-scalable, and resilient design; this ensures that the system remains operational in the presence of benign failures or resource depletion attacks, and that it dynamically *scales out*, or possibly *scales in*, according to request arrival rates. Our full-blown implementation on the Google Cloud Platform shows that deploying large-scale and efficient VPKI can be cost-effective: the processing latency to issue 100 pseudonyms is approximately 56 ms. More so, our experiments show that our VPKI system dynamically scales out or scales in according to the rate of pseudonyms requests. We formally assess the achieved security and privacy properties for the credential acquisition process. Overall, our scheme is a comprehensive solution that complements standards and can catalyze the deployment of secure and privacy-protecting VC systems.

1 Introduction

In Vehicular Communication (VC) systems, vehicles beacon Cooperative Awareness Messages (CAMs) and Decentralized Environmental Notification Messages (DENMs) periodically, at high rates [1], to enable transportation safety and efficiency. It has been well-understood that VC systems are vulnerable to attacks and that the privacy of their users is at stake. As a result, security and privacy solutions have been developed by standardization bodies (IEEE 1609.2 WG [2] and ETSI [3], [4], [5], [6]), harmonization efforts (C2C-CC [7]), and projects (SeVeCom [8], [9], [10], PRESERVE [11], and CAMP [12], [13], [14], [15], [16]). The de facto standard in VC systems to protect Vehicle-to-Vehicle (V2V)/Vehicle-to-Infrastructure (V2I) (V2X) communication is to use Public Key Cryptography (PKC) and pseudonymous authentication [3], [8], [9], [10], [15]: each vehicle is registered with one Long Term Certification Authority (LTCA), issuing its Long Term Certificate (LTC), and it is able to obtain a set of short-lived anonymized certificates, termed *pseudonyms*, from any Pseudonym Certification Authority (PCA), a pseudonym provider. Vehicles switch from one pseudonym to a non-previously used one, e.g., [3], [4], [5], [17], towards message unlinkability, as pseudonyms per se are inherently unlinkable. Pseudonymity is conditional, in the sense that the corresponding long-term vehicle identity can be retrieved by the Vehicular Public Key Infrastructure (VPKI) when needed, e.g., deviating from system policies.

Deploying a VPKI differs from a traditional Public Key Infrastructure (PKI), e.g., [18], [19], [20]. One of the most important factors is the VPKI dimension, i.e., the number of registered ''users'' (vehicles) and the multiplicity of certificates per user. According to the US Department of Transportation (DoT), a VPKI should be able to issue pseudonyms for more that 350 million vehicles across the Nation [21]. Considering the average daily commute time to be 1 hour [21] and a pseudonym lifetime of 5 minutes, the VPKI should be able to issue at least 1.5×10^{12} pseudonyms per year, i.e., 5 orders of magnitude more than the number of credentials the largest current PKI issues (10 million certificates per year [14]). Note that this number could be even greater for the entire envisioned Intelligent Transport Systems (ITSs) ecosystem, e.g., including pedestrians and cyclists, vehicular platooning, Location Based Services (LBSs) [3], [22], [23], and vehicular social networks [24]. These numbers could grow further if higher degrees of unlinkability, thus short pseudonym lifetimes, was chosen[1].

With emerging large-scale multi-domain VC environments [2], [3], [7], [22], [29], the efficiency of the VPKI and, more broadly, its scalability are paramount. Vehicles could request pseudonyms for a long period, e.g., 25 years [30]. However, extensive pre-loading with millions of pseudonyms per vehicle for a long period is computationally costly and inefficient in terms of utilization [31]. Moreover, in case of revocation [26], [27], [32], a huge CRL should be distributed among all vehicles due to the long lifespan of the credentials. In fact, sizable portion of any such oversized CRL is irrelevant to a receiving vehicle and can be left unused, i.e., wasting of significant bandwidth for CRL distribution [32], [33]. Alternatively, each vehicle could

• *The authors are with the Networked Systems Security Group, KTH Royal Institute of Technology, 10044 Stockholm, Sweden (e-mail: khodaei@kth.se; hnoroozi@kth.se; papadim@kth.se).*

1. The trend towards short-lived certificates is not unique to mobile systems, notably VC systems. It is also proposed for the classical Internet, although with less demanding requirements. Web server certificates are issued with a lifetime of a few days [25]. This essentially diminishes the need to distribute a large volume of revoked certificate after the latest Certificate Revocation List (CRL) was released [26], [27], [28].

interact with the VPKI regularly, e.g., once or a few times per day, not only to refill its pseudonym pool but also to fetch the latest revocation information[2]. However, the performance of a VPKI system can be drastically degraded and its operation undermined by a clogging Denial of Service (DoS) attack, targeting the VPKI [37], [31]: external adversaries with fake certificates or bogus tickets and internal adversaries with bogus Certificate Signing Requests (CSRs) could perform a signature flooding attack [38], thus, compromising the availability of the VPKI entities. Moreover, a *flash crowd* [39], e.g., a surge in pseudonym acquisition requests during rush hours, could render the VPKI unreachable, or drastically decrease its quality of service.

The cost of VPKI unavailability is twofold, affecting security (and consequently road safety) and privacy. An active malicious entity could prevent other vehicles from accessing the VPKI to fetch the latest revocation information. Moreover, signing CAMs with private keys corresponding to expired pseudonyms, or the LTC, is insecure and detrimental to user privacy. Thus, it is crucial to design a highly-available, scalable, and resilient VPKI that could efficiently issue pseudonyms in an *on-demand* fashion[3] [40], [41].

Considering a multi-domain deployment of VC systems, with a multiplicity of service providers, each vehicle could obtain pseudonyms from various service providers. The acquisition of multiple simultaneously valid (sets of) pseudonyms would enable an adversary to inject multiple erroneous messages, e.g., hazard notifications while the road conditions are safe, as if they were originated from multiple vehicles; or affect protocols based on voting, by sending out false, yet authenticated, information. Even though there are distributed schemes to identify Sybil [42] nodes, e.g., [43], [44], or secure and privacy-preserving VC systems can mitigate this vulnerability by relying on Hardware Security Modules (HSMs) [10], a VPKI system should prevent such credential misuse on the infrastructure side, e.g., [37], [31].

The VPKI entities are, often implicitly, assumed to be fully trustworthy. Given the experience from recent mobile applications, we need to guarantee strong user privacy even in the presence of honest-but-curious VPKI entities [29], [31]: they are honest, i.e., thoroughly complying with the best practices, specified protocols, and system policies, but curious, i.e., tempted to infer sensitive user information, thus harming user privacy. In the context of VC systems, no single VPKI entity should be able to link two successive pseudonyms belonging to the same vehicle. However, an honest-but-curious Resolution Authority (RA), responsible for resolving a pseudonym and identifying the long-term identity of a misbehaving vehicle, could harm user privacy by linking multiple sets of pseudonyms issued for a given vehicle[4]. This vulnerability results from the implicit binding of pseudonyms [45] (using a hash chain) towards an efficient distribution of the CRL [28], [32]. However, an RA could repeatedly conduct a pseudonym resolution process, e.g., by falsely claiming it received misbehaving reports. Beyond semi-honest VPKI entities that they are solely curious, a deviant LTCA could misbehave and issue fake *authorization tickets*, or issue multiple simultaneously valid tickets for a given vehicle; the tickets can be used later to obtain multiple sets of simultaneously valid pseudonyms. We address these challenges and re-define the pseudonym issuance process with a PCA and the registration process with an LTCA.

The recent trends towards cloudification with micro-service architecture provide high availability and dynamic scalability; the cloud-based VPKI system, e.g., [46], [47], could dynamically scale out, or possibly scale in, based on the VPKI as a Service (VPKIaaS)[5] system workload and the requests' arrival rate, so that it can comfortably handle *unexpected* demanding loads while being cost-effective by systematically allocating and deallocating resources. In cloud terminology, scaling in/out, termed *horizontal* scaling, refers to replicating a new instance of a service; while scaling up/down, termed *vertical* scaling, refers to allocating/deallocating resources for an instance of a given service. A malicious vehicle could repeatedly request pseudonyms; in fact, requests might be delivered to different replicas of a micro-service, releasing multiple simultaneously valid pseudonyms. Mandating a centralized database, shared among all replicas to ensure *isolation* and *consistency* of all transactions, would mitigate such a vulnerability. However, this contradicts highly efficient and timely pseudonyms provisioning for large-scale mobile systems.

Contributions: We leverage and *enhance* the state-of-the-art VPKI, and propose a *VPKIaaS* system towards a highly-available, dynamically-scalable, and fault-tolerant (highly-resilient) design, ensuring the system remains operational in the presence of benign failures or any resource depletion attack (clogging a DoS attack). At the same time, we re-define the vehicle registration process (with the LTCA) and improve our scheme to be resilient against a compromised LTCA. More so, we modify the pseudonym issuance process in two ways: (i) mitigating a DoS attack by internal malicious vehicles sending bogus CSRs, (ii) mitigating inferences an RA, that is honest-but-curious, could make and harm user privacy (when resolving two successive pseudonyms belong to the same vehicle). We present a formal security and privacy analysis of the pseudonym acquisition process in the presence of strong adversarial VPKI entities and vehicles. Moreover, our scheme eradicates Sybil-based misbehavior without diminishing the pseudonym acquisition efficiency. All procedures of deployment and migration to the cloud, e.g., bootstrapping phase, initializing the micro-services, pseudonym acquisition process, monitoring health and load metrics, etc., are fully automated. Through extensive experimental evaluation, we show that the VPKIaaS system could dynamically scale out, or possibly scale in. Our experimental evaluation shows a 36-fold improvement over prior work [47]: the processing delay to issue 100 pseudonyms for [47] is approximately 2000 ms, while it is approximately 56 ms in our system. Moreover, the performance of the VPKI

2. Note that Cellular-V2X can provide reliable and low-latency V2X communication with a wide range of coverage [34], [35], [36]; thus, network connectivity will not be a bottleneck.

3. Unlike issuing short-lived certificates [25] for the Internet that the certificates can be pre-generated and cached by the PKI, issuing on-demand pseudonyms for the VC system cannot be precomputed: each vehicle requests new pseudonymous certificate with a different public key, important for unlinkability (user privacy).

4. This vulnerability was not an issue in SECMACE [31] as distinct random numbers were used to implicitly bind the pseudonyms.

5. VPKIaaS refers to the cloudification of the VPKI and SECMACE+ is the system; VPKIaaS and SECMACE+ are used interchangeably.

system [31] drastically decreases when there is a surge in the pseudonym request arrival rates; on the contrary, our VPKIaaS system can comfortably handle demanding loads request while efficiently issuing batches of pseudonyms.

In the rest of the paper, we describe the system model and objectives (Sec. 2). We then explain the VPKIaaS system, detailing security protocols (Sec. 3), and providing a qualitative analysis (Sec. 4), followed by a quantitative analysis (Sec. 5) and related work (Sec. 6) before our conclusion (Sec. 7).

2 System Model and Objectives

2.1 Overview and Assumptions

A VPKI consists of a set of Certification Authorities (CAs) with distinct roles: the Root CA (RCA), the highest-level authority, certifies other lower-level authorities; the LTCA is responsible for the vehicle registration and the Long Term Certificate (LTC) issuance, and the PCA issues pseudonyms for the registered vehicles. Pseudonyms have a lifetime (a validity period), typically ranging from minutes to hours; in principle, the shorter the pseudonym lifetime is, the higher the unlinkability and thus the higher privacy protection can be achieved. We assume that each vehicle is registered only with its *Home-LTCA (H-LTCA)*, the *policy decision and enforcement point*, reachable by the registered vehicles. Without loss of generality, a *domain* can be defined as a set of vehicles in a region, registered with the H-LTCA, subject to the same administrative regulations and policies [29], [48]. There can be several PCAs, each active in one or more domains; any legitimate, i.e., registered, vehicle is able to obtain pseudonyms from any PCA, the pseudonym provider (as long as there is a trust established between the two CAs). Trust between two domains can be established with the help of the RCA, or through cross certification.

Each vehicle interacts with the VPKI entities to obtain a batch of pseudonyms, each having a corresponding short-term private key, to sign and disseminate their mobility information, e.g., CAMs or DENMs, time- and geo-stamped, periodically or when needed as a response to a specific event. Fig. 1 shows an overview of a VPKI with three domains, A, B and C. Domains A and B have established trust with the help of a higher level authority, i.e., the RCA, while domains B and C have established security association by cross certification. The vehicles in the figure are labeled with the domains they are affiliated to. A vehicle registered in domain A digitally signs outgoing messages with the private key, k_v^i, corresponding to the pseudonym P_v^i, which signifies the current valid pseudonym signed by the PCA. The pseudonym is then attached to the signed messages to enable verification by any recipient. Upon reception, the pseudonym is verified before the message itself (signature validation). This process ensures communication authenticity, message integrity, and non-repudiation. Vehicles switch from one pseudonym to another one (non-previously used) to achieve unlinkability, thus protecting sender's privacy as the pseudonyms are inherently unlinkable.

Each vehicle *"decides"* when to trigger the pseudonym acquisition process based on various factors [41]. Such a scheme requires sparse connectivity to the VPKI, but it facilitates an On-Board Unit (OBU) to be *preloaded* with pseudonyms proactively, covering a longer period, e.g., a

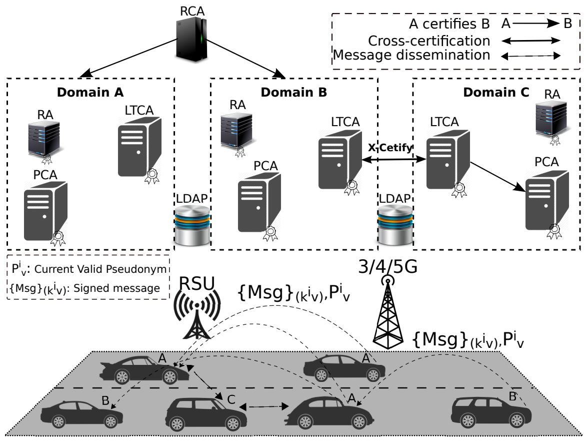


Fig. 1: A VPKI Overview for Multi-domain VC Systems.

week or a month, should the connectivity be expected heavily intermittent. A universally fixed interval, Γ, is specified by the H-LTCA and all pseudonyms in that domain are issued with the lifetime (τ_P) aligned with the VPKI clock [31]. As a result of this policy, at any point in time, all the vehicles transmit using pseudonyms that cannot be distinguished based on their issuance time thanks to this time alignment.

All vehicles (OBUs) registered are equipped with HSMs, ensuring that private keys never leave the HSM. We assume that there is a misbehavior detection system, e.g., [49], that triggers revocation. The Resolution Authority (RA) can initiate a process to resolve and revoke all pseudonyms of a misbehaving vehicle [50]: it interacts with the corresponding PCAs and LTCA to resolve and revoke all credentials issued for a misbehaving vehicle. Consequently, the misbehaving vehicle can no longer obtain credentials from the VPKI. The VPKI is responsible for distributing the CRLs and notifying all legitimate entities about the revocation, e.g., [28], [32]. We further assume that the cloud service providers are honest and they provide a service with the desired Service Level Agreement (SLA); in terms of secret management, we assume that the cloud service providers are fully trustworthy. For a detailed description of secret management in the cloud, we refer interested readers to our earlier publication [45].

2.2 Adversarial Model and Requirements

We first extend the general adversary model in secure vehicular communications [31], [48] to include an *honest-but-curious* service provider, i.e., a VPKI entity that attempts to gain advantages towards its goal, e.g., profiling users. In addition, malicious PCAs could try to (i) issue multiple sets of (simultaneously valid) pseudonyms for a legitimate vehicle, or (ii) issue a set of pseudonyms for a non-existing (illegitimate) vehicle, or (iii) fraudulently accuse different vehicles (users) during a pseudonym resolution process. A deviant LTCA could attempt to map a different LTC during the resolution process, thus misleading the system. Furthermore, a deviant LTCA could unlawfully register illegitimate vehicles, i.e., issue fake LTCs, fake *authorization tickets* (to be used during pseudonym acquisition process), or multiple simultaneously valid tickets for a given vehicle. A semi-honest RA can also continuously initiate pseudonym validation process towards inferring user sensitive information, e.g., linking successive pseudonyms belonging to the same vehicle. Multiple VPKI servers can collude, i.e., share information that each of them individually infers with the others, to harm user privacy.

In a multi-PCA environment, malicious (compromised) clients, i.e., internal adversaries, raise two challenges. First, they could repeatedly request multiple simultaneously valid pseudonyms, thus preparing to act (and misbehave), each posing as multiple registered legitimate-looking vehicles. Second, they could degrade the system operations by mounting a clogging DoS attack against the VPKI servers. *External adversaries*, i.e., unauthorized entities, could try to harm the system operations by launching a DoS (or a Distributed DoS (DDoS)) attack, thus degrading the availability of the system. But they are unable to successfully forge messages or 'crack' the employed cryptosystems and cryptographic primitives. *Internal adversaries* could aggressively request tickets with fake certificates or pseudonyms with bogus ticket. Alternatively, they could send pseudonym requests with forge signature on the CSRs; in the worst case (for the PCA), a vehicle provides valid signature in all the CSRs, but the last one, i.e., bogus signature only in the latest CSR when requesting multiple pseudonyms. This requires the PCA to validate the signature of all CSR until it detects the last one with bogus signature.

Security and privacy requirements for V2X communications have been specified in [48], and additional requirements for VPKI entities and the CRL distribution in [31], [32]. Beyond the aforementioned requirements, we need to thwart Sybil-based attacks in a cloud-based VPKIaaS system (without degrading efficient pseudonym issuance). At the same time, we need to ensure that the VPKIaaS system is dynamically scalable, i.e., that the system *dynamically* scales out, or possibly scales in, according to the requests' arrival rate, to handle any demanding load while being cost-effective by systematically allocating and deallocating resources. Moreover, we need to ensure that the scheme is resilient to any resource depletion attack.

3 Service Overview & Security Protocols

In the registration phase, each H-LTCA registers vehicles within its domain and maintains their long-term identities. At the bootstrapping phase, each vehicle needs to discover the VPKI-related information, e.g., the available PCAs in its home domain, or the desired Foreign-LTCA (F-LTCA) and PCAs in a foreign domain, along with their corresponding certificates. To facilitate the overall intra- and multi-domain operations, a vehicle first finds such information from a Lightweight Directory Access Protocol (LDAP) [51] server. This is carried out without disclosing the long-term identity of the vehicle. We presume connectivity to the VPKI, e.g., via Roadside Units (RSUs) or Cellular-V2X; should the connectivity be intermittent, the vehicle, i.e., the OBU, could initiate pseudonym provisioning proactively based on different parameters, e.g., the number of remaining valid pseudonyms and the residual trip duration.

The H-LTCA authenticates and authorizes vehicles over a unidirectional (server-only) authenticated Transport Layer Security (TLS)/Secure Sockets Layer (SSL) [52] tunnel. This way the vehicle obtains a *native ticket* (n-tkt) from its H-LTCA while the targeted PCA or the actual pseudonym acquisition period is hidden from the H-LTCA; the ticket is anonymized and it does not reveal the vehicle identity (Protocol 2 and Protocol 3). The ticket is then presented to the intended PCA, over a unidirectional (server-only) authenticated TLS, to obtain pseudonyms (Protocol 4).

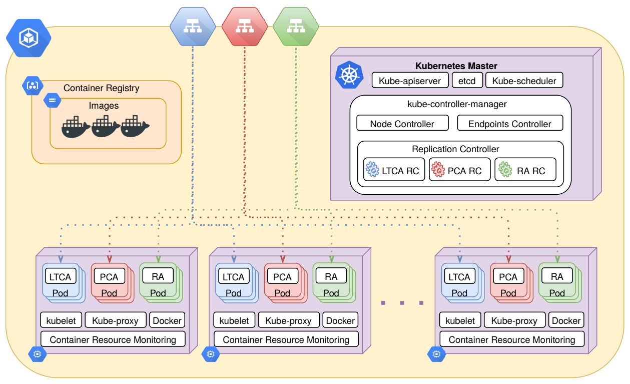


Fig. 2: A High-level Overview of VPKIaaS Architecture.

When the vehicle travels in a foreign domain, it should obtain new pseudonyms from a PCA operating in that domain; otherwise, the vehicle would stand out and be more easily traceable. The vehicle first requests a *foreign ticket* (f-tkt) from its H-LTCA (without revealing its targeted F-LTCA) so that the vehicle can be authenticated and authorized by the F-LTCA. In turn, the F-LTCA provides the vehicle with a new ticket (n-tkt), which is native within the domain of the F-LTCA to be used for pseudonym acquisition in that (foreign) domain. The vehicle then interacts with its desired PCA to obtain pseudonyms. Obtaining an f-tkt is transparent to the H-LTCA: the H-LTCA cannot distinguish between native and foreign ticket requests. This way, the PCA in the foreign domain cannot distinguish native requesters from foreign ones. For liability attribution, our scheme enables the RA, with the help of the PCA and the LTCA, to initiate a resolution process, i.e., to resolve a pseudonym to its long-term identity. Each vehicle can interact with any PCA, within its home or a foreign domain, to fetch the CRL [32] and perform Online Certificate Status Protocol (OCSP) [37] operations, authenticated with a current valid pseudonym.

3.1 VPKI as a Service (VPKIaaS)

We migrate the VPKI on the Google Cloud Platform (GCP) [53] for the availability, reliability, and dynamic scalability of the VPKI system under various circumstances. Fig. 2 illustrates a high-level abstraction of the VPKIaaS architecture on a managed Kubernetes cluster [54] on GCP. The RCA is assumed to be off-line, not included in this abstraction. A set of Pods will be created for each micro-service, e.g., LTCA[6] or PCA, from their corresponding container images, facilitating their horizontal scalability. When the rate of pseudonym requests increases, the Kubernetes master, shown on the top in Fig. 2, schedules new Pods or kills a running Pod in case of benign failures, e.g., system faults or crashes, or resource depletion attacks, e.g., a DoS attack. The number of Pods could be scaled out to a pre-set number, defined in the deployment configuration, or scaled out to the amount of available resources enabled by Kubernetes nodes.

6. The terms LTCA and H-LTCA are used interchangeably.

TABLE 1: Notation Used in the Protocols.

$AddToCRL(LTC)$	Adding a pseudonym to the CRL
CK	Commitment Key
CN	Common Name
$ExtractPublicKey(LTC_v)$	Extracting the public key from the LTC
$GenRnd()$	Generating a random number
$H'(), H()$	Secure Hash Functions
$Id_{req}, Id_{res}, Id_{ca}$	Request, Response, CA Unique Identifiers
(K_v^i, k_v^i)	Pseudonymous public/private key pairs
(LK_v, Lk_v)	Long-term public/private key pairs
LTC	Long Term Certificate
$(msg)_{\sigma_v}$	A signed message with the vehicle's private key; σ_v represents the signature of the message
N, Rnd	Nonce, A Random Number
$n\text{-}tkt, tkt$	Native Ticket
$(P_v^i)_{pca}$, P_v^i	A pseudonym signed by the PCA
$RedisQuery(SN)$	Checking if a serial number exists in Redis
$RedisUpdate(SN, value)$	Updating a SN (key) with *value* in the Redis
$Resolve(tkt)$	Resolving a ticket to its LTC
$Sign(Lk, msg)$	Signing a message with the private key (Lk)
SN	Serial Number
t_{now}, t_s, t_e	Current, starting, and ending timestamps
V	Vehicle
$Verify(LK, msg)$	Verifying a message with the public key (LK)
τ_P	Pseudonym Lifetime
ζ, χ	Temporary Variables
Γ	Interacting interval with the VPKI

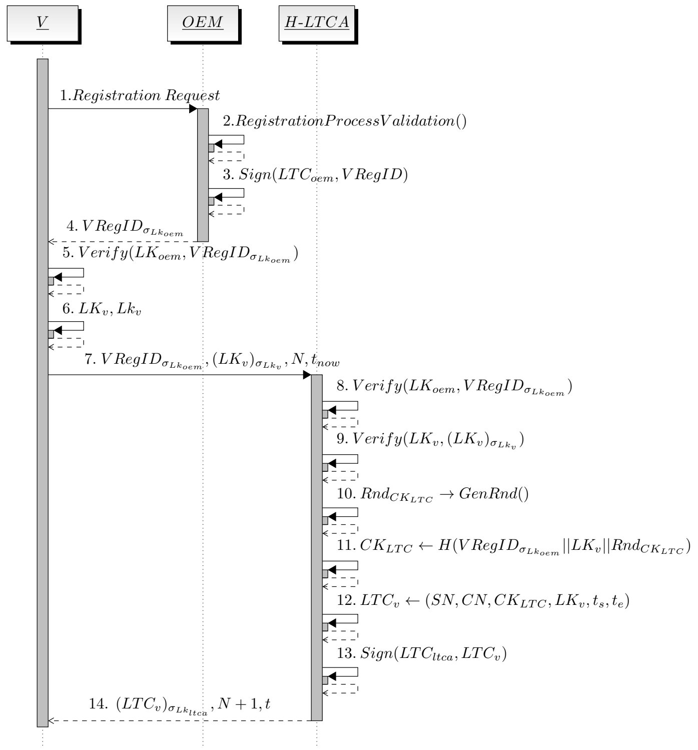
Each Pod publishes two types of metrics: *load* and *health*. The load metric values are generated by a resource monitoring service, which facilitates horizontal scaling of a micro-service: upon reaching a threshold of a pre-defined load, replication controller replicates a new instance of the micro-service to ensure a desired SLA. The health metric ensures correct operation of a micro-service by persistently monitoring its status: a faulty Pod is killed and a new one is created. In our VPKIaaS system, we define CPU usage as the load metric. In order to monitor the health condition of a micro-service, dummy requests (dummy tickets for the LTCA micro-services and dummy pseudonyms for the PCA micro-services) are locally queried by each Pod. A dummy ticket request is constructed by an LTCA Pod to validate the correctness of the ticket issuance procedure, while a dummy pseudonym request is constructed by a PCA Pod to ensure the correctness of pseudonym issuance procedure. Such dummy requests cannot be used by a compromised Pod to issue fake pseudonyms (see Sec. 4).

3.2 Security Protocols

We describe the pseudonym acquisition (Protocol 4) and pseudonym issuance validation processes (Protocol 5) to identify a misbehaving PCA issuing fraudulent pseudonym. Furthermore, in order to mitigate Sybil attacks on the side of VPKIaaS system, we propose two protocols (Protocols 7 and 8): an in-memory key-value Redis database [55] is shared among all replicas of a micro-service, to facilitate efficient validation of tickets and pseudonyms requests. Table 1 shows the notation used in the security protocols.

Vehicle Registration and LTC Update (Protocol 1)

Each vehicle first interacts with its Original Equipment Manufacturer (OEM) (or the department of transportation) to initiate the registration process (step 1.1, i.e., step 1 in Protocol 1). To validate the registration process (step 1.2), the OEM provides a signed registration coupon, $VRegID_{\sigma_{Lk_{oem}}}$ (step 1.3). Upon receiving the coupon (step 1.4), the vehicle verifies the signature (step 1.5), generates a pair of public and private keys (step 1.6), and prepares the CSR [56]. The vehicle then sends the LTC acquisition request to the H-LTCA (step 1.7).



V
OEM
H-LTCA
1.Registration Request
2.RegistrationProcessValidation()
3. Sign(LTC_oem, VRegID)
4. VRegID_σLk_oem
5. Verify(LK_oem, VRegID_σLk_oem)
6. LK_v, Lk_v
7. VRegID_σLk_oem, (LK_v)_σLk_v, N, t_now
8. Verify(LK_oem, VRegID_σLk_oem)
9. Verify(LK_v, (LK_v)_σLk_v)
10. Rnd_CK_LTC → GenRnd()
11. CK_LTC ← H(VRegID_σLk_oem||LK_v||Rnd_CK_LTC)
12. LTC_v ← (SN, CN, CK_LTC, LK_v, t_s, t_e)
13. Sign(LTC_ltca, LTC_v)
14. (LTC_v)_σLk_ltca, N+1, t

Protocol 1: Vehicle Registration with the OEM and the LTC Acquisition from the H-LTCA.

Having received a request, the H-LTCA verifies the coupon signed by the OEM, assuming that trust is established between the two entities (step 1.8). It then initiates a proof-of-possession protocol for the ownership of the corresponding private keys (step 1.9). Upon successful verification, the H-LTCA generates a random number, $Rnd_{CK_{LTC}}$ (step 1.10). Then, it calculates the LTC's commitment key, CK_{LTC}: $H(VRegID_{\sigma_{Lk_{oem}}}||LK_v||Rnd_{CK_{LTC}})$ (step 1.11). This essentially prevents a deviant LTCA from unlawfully registering an illegitimate vehicle in the system. The LTCA then integrates the CK_{LTC} in the LTC and issues the certificates (steps 1.12–1.13). Finally, the vehicle validates the signature on the LTC (step 1.14).

Ticket Acquisition Process (Protocols 2 and 3)

Assume the OBU decides to obtain pseudonyms from a specific PCA. It first interacts with its H-LTCA to obtain a valid ticket. To conceal the actual identity of its desired PCA from the LTCA, it calculates the hash value of the concatenation of the specific PCA identity with a random number[7] (steps 2.1–2.2). The vehicle prepares the request and signs it with the private key corresponding to its LTC (step 2.3–2.4) before returning the ticket request (step 2.5). It will then interact with the LTCA over a unidirectional (server-only) authenticated TLS [52] tunnel.

7. The storage cost for these random numbers is reasonably cheap, e.g., 264 million vehicles with average trip duration of 1 hour require 32 GB per day (25$ per month, US multi-region [57]).

Protocol 2 Ticket Request from the LTCA (by the Vehicle)

```
1: procedure ReqTicket($Id_{PCA}, t_s, t_e$)
2:     $Rnd_{tkt} \leftarrow GenRnd()$
3:     $\zeta \leftarrow (Id_{req}, H(Id_{PCA} \| Rnd_{tkt}), t_s, t_e)$
4:     $(msg)_{\sigma_v} \leftarrow Sign(Lk_v, \zeta)$
5:     return $((msg)_{\sigma_v}, \text{LTC}_v, N, t_{now})$
6: end procedure
```

Protocol 3 Issuing a Ticket (by the LTCA)

```
1: procedure IssueTicket($(msg)_{\sigma_v}, \text{LTC}_v, N, t_{now}$)
2:     Verify($\text{LTC}_v, (msg)_{\sigma_v}$)
3:     $Rnd_{CK_{n\text{-}tkt}} \leftarrow GenRnd()$
4:     $CK_{n\text{-}tkt} \leftarrow H(\text{LTC}_v \| t_s \| t_e \| Rnd_{CK_{n\text{-}tkt}})$
5:     $\zeta \leftarrow (SN, H(Id_{PCA} \| Rnd_{n\text{-}tkt}), CK_{n\text{-}tkt}, t_s, t_e, Exp_{n\text{-}tkt})$
6:     $(n\text{-}tkt)_{\sigma_{ltca}} \leftarrow Sign(Lk_{ltca}, \zeta)$
7:     return $(Id_{res}, (n\text{-}tkt)_{\sigma_{ltca}}, Rnd_{CK_{n\text{-}tkt}}, N+1, t_{now})$
8: end procedure
```

Upon reception of the ticket request, the LTCA verifies the LTC (thus authenticating and authorizing the requester) and the signed message (step 3.2). The LTCA generates a random number ($Rnd_{CK_{tkt}}$) and calculates the *ticket commitment key* (CK_{tkt}) to bind the ticket to the LTC as: $H(\text{LTC}_v \| t_s \| t_e \| Rnd_{CK_{tkt}})$ (steps 3.3–3.4); this prevents a compromised LTCA from mapping a different LTC during the resolution process. The LTCA then encapsulates (step 3.5), signs (step 3.6), and delivers the response (step 3.7).

3.2.1 *Pseudonym Acquisition Process (Protocol 4)*

Each vehicle first requests an anonymous ticket from its H-LTCA, using it to interact with the desired PCA to obtain pseudonyms. Upon reception of a valid ticket, it generates CSRs with Elliptic Curve Digital Signature Algorithm (ECDSA) public/private key pairs [2], [3] and sends the request to the PCA. The vehicle-LTCA and vehicle-PCA communications are over a unidirectional (server-only) authenticated TLS [52] (or Datagram TLS (DTLS) [58]); this ensures the PCA does not infer the actual requester identity.

Having received a request, the PCA verifies the ticket signed by the H-LTCA (assuming trust is established between the two) (steps 4.2–4.3). The PCA then decapsulates the ticket and verifies the pseudonym provider identity (step 4.4–4.5). Then, the PCA generates a random number (step 4.6) and initiates a proof-of-possession protocol to verify the ownership of the corresponding private keys by the vehicle (step 4.9). Then, it calculates the *commitment key*, $CK_{P_v^i}$, as: $H(CK_{tkt} \| K_v^i \| t_s^i \| t_e^i \| H'(H^i(Rnd_{psnym})))$ (step 4.10). This essentially prevents a compromised PCA from mapping a different ticket during the resolution process, or it identifies a malicious PCA that issued a pseudonym without a valid ticket provided by the client (requester). The PCA implicitly correlates a batch of pseudonyms belonging to each requester (steps 4.11–4.15). This essentially enables efficient distribution of the CRL [32]: the PCA only needs to include one entry per batch of pseudonyms without compromising their unlinkability. Finally, the PCA issues the pseudonyms by signing them using its private key (steps 4.16–4.17) and delivers the response (step 4.19).

An RA could harm user privacy by frequently initiating the pseudonym resolution protocol towards linking multiple sets of pseudonyms issued for a given vehicle. This vulnerability is the result of the implicit binding of pseudonyms (using a hash chain) issued to a given requester per an interval (Γ) [32], [45]. An RA could either repeatedly conduct a pseudonym resolution process, e.g., by falsely claiming it received misbehaving reports, or perform a pseudonym validation process for a victim vehicle towards linking all its successive pseudonyms; this can be conducted if an RA colludes with an external adversary, eavesdropping the transcript of the VC system.

Protocol 4 Issuing Pseudonyms (by the PCA)

```
1: procedure IssuePsnyms($Req$)
2:     $Req \rightarrow (Id_{req}, Rnd_{n\text{-}tkt}, tkt_{\sigma_{ltca}},$
           $\{(K_v^1, \dots, K_v^n)_{\sigma_{k_v^1}}, \cdots, (K_v^1, \dots, K_v^n)_{\sigma_{k_v^n}}\}, N, t_{now})$
3:     Verify($\text{LTC}_{ltca}, (tkt)_{\sigma_{ltca}}$)
4:     $tkt_{\sigma_{ltca}} \rightarrow (SN, H(Id_{PCA} \| Rnd_{tkt}), CK_{tkt}, t_s, t_e, Exp_{tkt})$
5:     $H(Id_{this\text{-}pca} \| Rnd_{n\text{-}tkt}) \overset{?}{=} H(Id_{pca} \| Rnd_{n\text{-}tkt})$
6:     $Rnd_{psnym} \leftarrow GenRnd()$
7:     for i:=1 to n do
8:         Begin
9:             Verify($K_v^i, (K_v^1, \dots, K_v^n)_{\sigma_{k_v^i}}$)
10:            $CK_{P_v^i} \leftarrow H(CK_{tkt} \| K_v^i \| t_s^i \| t_e^i \| H'(H^i(Rnd_{psnym})))$
11:            if $i = 1$ then
12:                $SN^i \leftarrow H(CK_{P_v^i} \| H'(H^i(Rnd_{psnym})))$
13:            else
14:                $SN^i \leftarrow H(SN^{i-1} \| H'(H^i(Rnd_{psnym})))$
15:            end if
16:            $\zeta \leftarrow (SN^i, K_v^i, CK_{P_v^i}, t_s^i, t_e^i)$
17:            $(P_v^i)_{\sigma_{pca}} \leftarrow Sign(Lk_{pca}, \zeta)$
18:        End
19:    return $(Id_{res}, \{(P_v^1)_{\sigma_{pca}}, \dots, (P_v^n)_{\sigma_{pca}}\}, Rnd_{psnym}, N{+}1, t_{now})$
20: end procedure
```

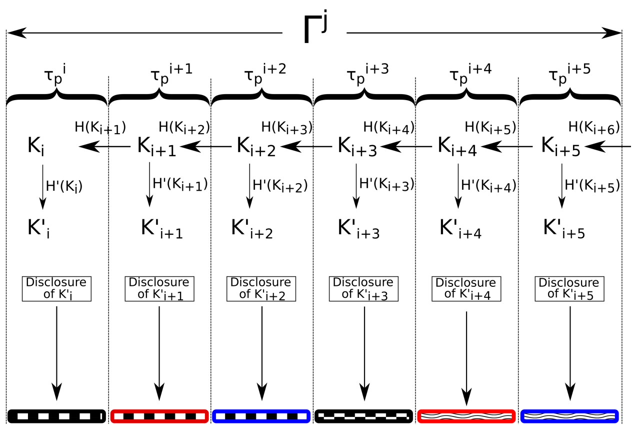


Fig. 3: Issuing Multiple Pseudonyms in a Γ Interval.

Fig. 3 shows how to mitigate this vulnerability: the PCA creates a hash chain[8], N times, for each Γ^j, where N is $\frac{\Gamma^j}{\tau_p}$ and it assigns the hash values sequentially to the time intervals τ_P^i, i.e., one hash value per τ_P (step 4.10 in Protocol 4). The PCA calculates a new hash function, $H'(K)$; each hash value is used to calculate the *commitment key*, which is then integrated into a pseudonym and signed by the PCA. In case the RA needs to perform pseudonym validation process for a single pseudonym, e.g., whose lifetime falls within τ_P^{i+5}, the PCA would disclose K'_{i+5};

8. Unlike using the hash chain for lightweight authentication, we leverage the hash chain for efficient revocation [28], [32] and improved user privacy protection.

this prevents the RA from linking the pseudonym to its successive ones (as it is infeasible to derive K_{i+5} from K'_{i+5}). Still, the RA could initiate a process to revoke all pseudonyms belonging to a malfunctioning or malicious vehicle. Thus, the PCA discloses K_{i+5}: this enables the RA to simply derive subsequent pseudonyms from the K_{i+5}. Note that performing hash operations are very efficient and this does not impose excessive overhead to the pseudonym issuance process: calculating one hash operation would result in 0.00069 ms, i.e., 0.18% extra overhead in issuing 100 pseudonyms.

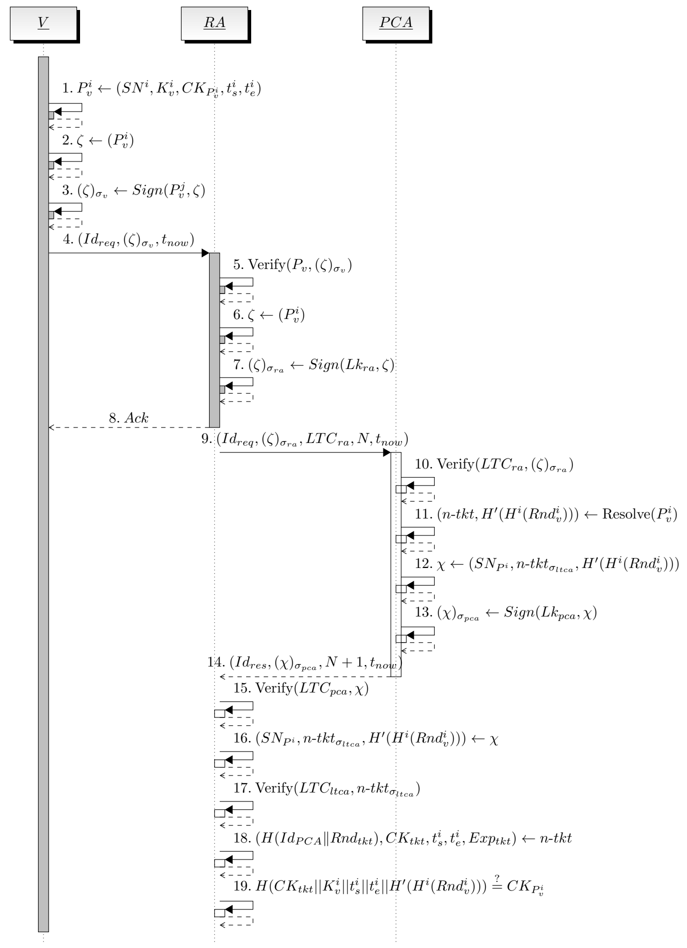
DoS-resilient CSR verification: According to the standard [59], each vehicle needs to provide the proof of possession in each CSR to prove it has the private key corresponding to each requested public key. The VPKI system should validate proof of possession for each public key. However, this could compromise on-demand pseudonym acquisition: adversarial vehicles construct legitimate-looking CSRs with bogus proof-of-possession signature for the last one. This results in a *benign* failure but it imposes excessive computation overhead on the PCA, thus diminishing the operation of the system[9]. We mitigate this without imposing extra overhead on the vehicle or the PCA side: rather than providing the proof of possession for one public key in each CSR ($(K_v^1)_{\sigma_{k_v^1}}, \dots, (K_v^n)_{\sigma_{k_v^n}}$), each vehicle provides the proof of possession for all public keys in each CSR ($(K_v^1, \dots, K_v^n)_{\sigma_{k_v^1}}, \cdots, (K_v^1, \dots, K_v^n)_{\sigma_{k_v^n}}$). As a result, each vehicle signs the hash of all public keys with all the private keys (steps 4.2 and 4.9 in Protocol 4). With this modification, the PCA could randomly opt in one CSR to validate the proof-of-possession for all CSRs as the requester non-repudiably proved it possessed the private keys for all public keys. Note that the changes in the CSR process do not impose extra overhead to the overall pseudonym acquisition process: rather than performing a hash operation on one public key, a vehicle calculates a hash on all public keys. Moreover, there is no additional communication overhead to the CSRs since a hash function maps any arbitrary size to a fixed-sized value.

3.2.2 *Pseudonym Issuance Validation (Protocol 5)*

Upon receiving a request for misbehavior identification, e.g., multiple suspicious traffic congestion alerts sent to a traffic monitoring system, an entity could send a request to the RA to validate the pseudonym issuance process of a *suspicious* pseudonym (step 5.1– 5.4). The RA validates the request and interacts with the corresponding PCA that issued the pseudonym, to provide evidence for the pseudonym issuance procedure; in fact, this process ensures that an actual vehicle requested the pseudonym by providing a valid ticket, also guarantees the PCA did not issue a pseudonym for an illegitimate vehicle (step 5.5– 5.9).

Upon receipt, the PCA validates the request and provides the corresponding ticket and $Rnd_{CK_{P_v^i}}$, used to issue the pseudonym. The PCA response is signed and sent back to the RA (step 5.10– 5.14). In turn, the RA verifies it, enables validating the ticket using the public key of the LTCA, and checks $H(CK_{tkt}||K_v^i||t_s^i||t_e^i||H'(H^i(Rnd_v^i))) \overset{?}{=} CK_{P_v^i}$

9. Built-in DoS attack mitigation techniques in the GCP cannot mitigate such an attack as the requests originated from legitimate requesters, holding valid tickets.

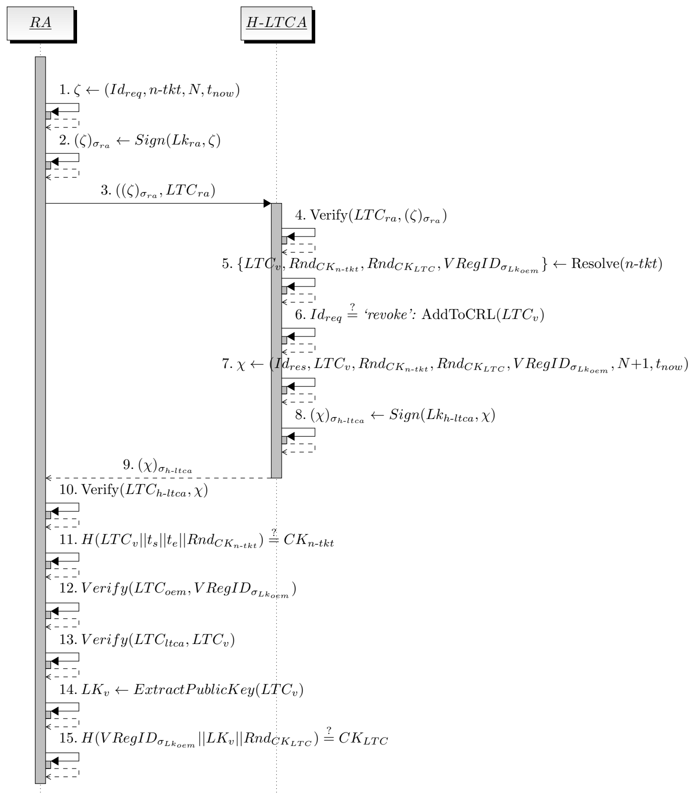

$\underline{V}$
$\underline{RA}$
$\underline{PCA}$
1. $P_v^i \leftarrow (SN^i, K_v^i, CK_{P_v^i}, t_s^i, t_e^i)$
2. $\zeta \leftarrow (P_v^i)$
3. $(\zeta)_{\sigma_v} \leftarrow Sign(P_v^j, \zeta)$
4. $(Id_{req}, (\zeta)_{\sigma_v}, t_{now})$
5. Verify$(P_v, (\zeta)_{\sigma_v})$
6. $\zeta \leftarrow (P_v^i)$
7. $(\zeta)_{\sigma_{ra}} \leftarrow Sign(Lk_{ra}, \zeta)$
8. Ack
9. $(Id_{req}, (\zeta)_{\sigma_{ra}}, LTC_{ra}, N, t_{now})$
10. Verify$(LTC_{ra}, (\zeta)_{\sigma_{ra}})$
11. $(n\text{-}tkt, H'(H^i(Rnd_v^i))) \leftarrow$ Resolve(P_v^i)
12. $\chi \leftarrow (SN_{P^i}, n\text{-}tkt_{\sigma_{ltca}}, H'(H^i(Rnd_v^i)))$
13. $(\chi)_{\sigma_{pca}} \leftarrow Sign(Lk_{pca}, \chi)$
14. $(Id_{res}, (\chi)_{\sigma_{pca}}, N+1, t_{now})$
15. Verify(LTC_{pca}, χ)
16. $(SN_{P^i}, n\text{-}tkt_{\sigma_{ltca}}, H'(H^i(Rnd_v^i))) \leftarrow \chi$
17. Verify$(LTC_{ltca}, n\text{-}tkt_{\sigma_{ltca}})$
18. $(H(Id_{PCA}\|Rnd_{tkt}), CK_{tkt}, t_s^i, t_e^i, Exp_{tkt}) \leftarrow n\text{-}tkt$
19. $H(CK_{tkt}||K_v^i||t_s^i||t_e^i||H'(H^i(Rnd_v^i))) \overset{?}{=} CK_{P_v^i}$

Protocol 5: Pseudonym Issuance Validation.

(steps 5.15– 5.19). If the hash calculation results in the same hash values, it confirms that the pseudonym has been issued based on a valid ticket, i.e., properly issued by the LTCA. Moreover, it ensures the PCA could not have issued the pseudonym for a *non-existing* vehicle. Note that upon performing pseudonym issuance validation process, the actual identity of a vehicle is not disclosed, i.e., user privacy is strongly protected.

3.2.3 *Ticket Issuance Validation, and LTC Resolution and Revocation Processes (Protocol 6)*

The RA queries the corresponding H-LTCA to validate the ticket issuance process, corresponding a suspicious pseudonym and to have the vehicle identity identified, i.e., resolving the LTC of the vehicle. Further, the RA could request to evict a misbehaving or malfunctioning vehicle from the system; it would request the LTCA and the PCA to revoke the LTC and all valid pseudonyms by adding them to the CRL [28], [32]. The RA prepares and signs a message and sends the request to the H-LTCA (steps 6.1-6.3). Upon request verification, the H-LTCA resolves the $n\text{-}tkt$, requested by the RA (steps 6.4-6.5). The LTCA returns $\{LTC_v, Rnd_{CK_{n\text{-}tkt}}, Rnd_{CK_{LTC}}, VRegID_{\sigma_{Lk_{oem}}}\}$. The $Rnd_{CK_{n\text{-}tkt}}$ enables validating $CK_{n\text{-}tkt}$; the $Rnd_{CK_{LTC}}$ and $VRegID_{\sigma_{Lk_{oem}}}$ enable validating the commitment key CK_{LTC}. In case of revocation request, the H-LTCA revokes the LTC and adds it to the CRL (step 6.6). The H-LTCA


RA
H-LTCA
1. $\zeta \leftarrow (Id_{req}, n\text{-}tkt, N, t_{now})$
2. $(\zeta)_{\sigma_{ra}} \leftarrow Sign(Lk_{ra}, \zeta)$
3. $((\zeta)_{\sigma_{ra}}, LTC_{ra})$
4. Verify$(LTC_{ra}, (\zeta)_{\sigma_{ra}})$
5. $\{LTC_v, Rnd_{CK_{n\text{-}tkt}}, Rnd_{CK_{LTC}}, VRegID_{\sigma_{Lk_{oem}}}\} \leftarrow$ Resolve$(n\text{-}tkt)$
6. $Id_{req} \overset{?}{=}$ *'revoke'*: AddToCRL(LTC_v)
7. $\chi \leftarrow (Id_{res}, LTC_v, Rnd_{CK_{n\text{-}tkt}}, Rnd_{CK_{LTC}}, VRegID_{\sigma_{Lk_{oem}}}, N{+}1, t_{now})$
8. $(\chi)_{\sigma_{h\text{-}ltca}} \leftarrow Sign(Lk_{h\text{-}ltca}, \chi)$
9. $(\chi)_{\sigma_{h\text{-}ltca}}$
10. Verify$(LTC_{h\text{-}ltca}, \chi)$
11. $H(LTC_v||t_s||t_e||Rnd_{CK_{n\text{-}tkt}}) \overset{?}{=} CK_{n\text{-}tkt}$
12. $Verify(LTC_{oem}, VRegID_{\sigma_{Lk_{oem}}})$
13. $Verify(LTC_{ltca}, LTC_v)$
14. $LK_v \leftarrow ExtractPublicKey(LTC_v)$
15. $H(VRegID_{\sigma_{Lk_{oem}}}||LK_v||Rnd_{CK_{LTC}}) \overset{?}{=} CK_{LTC}$

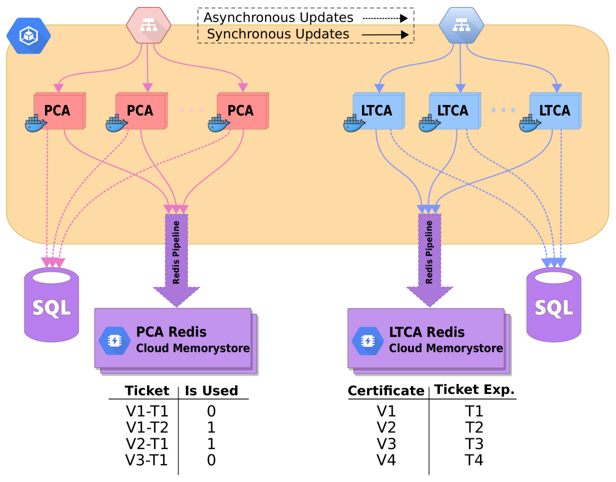
Protocol 6: Ticket Issuance Validation, and LTC Resolution and Revocation.

prepares a response, signs, and delivers it to the RA (steps 6.7-6.9). The RA verifies the response signature and confirms if the H-LTCA mapped the ticket to the correct LTC$_v$ by validating the $CK_{n\text{-}tkt}$ (steps 6.10-6.11). Furthermore, the RA verify the $VRegID_{\sigma_{Lk_{oem}}}$ and the LTC_v (steps 6.12-6.13), and extracts the public key (step 6.14). Finally, the RA confirms if the H-LTCA has mapped LTC to the correct $VRegID_{\sigma_{Lk_{oem}}}$ by validating the CK_{LTC} (step 6.15).

3.3 Mitigating Sybil Attacks on the VPKIaaS System

Multiple replicas of a micro-service interact with the same database to accomplish their operations; all replicas of LTCAs should interact with the database to store information about tickets they issue. The same way, all replicas of PCAs interact with a single database to validate an authorization ticket and store information corresponding to issued pseudonyms. Micro-services could opt in to utilize their shared MySQL database either synchronously or asynchronously[10]. Asynchronous interaction of the micro-services and the shared database would result in efficient pseudonym issuance. However, a malicious vehicle could repeatedly submit requests. If the micro-services do not synchronously validate tickets and pseudonym requests, one can obtain multiple sets of pseudonyms in the event such multiple requests were delivered to different replicas. On the other hand, synchronous interaction of the micro-services

10. A synchronous interaction with a database implies enforcing limits on accessing to a resource by locking it to ensure the consistency of all transactions. An asynchronous interaction, though, implies that requests are proceeded without waiting to complete a transaction; the execution will happen later via an asynchronous callback function.



Ticket	Is Used
V1-T1	0
V1-T2	1
V2-T1	1
V3-T1	0

Certificate	Ticket Exp.
V1	T1
V2	T2
V3	T3
V4	T4

Fig. 4: VPKIaaS Sybil mitigation with Redis and MySQL.

and the shared database would prevent issuing multiple sets of pseudonym for a given requester, thus, eradicating Sybil-based misbehavior. However, this would drastically degrade system performance, notably timely on-demand issuance of pseudonyms. The performance of the relational database, e.g., MySQL, used in [31], can be highly degraded by synchronized interactions, e.g., [60]. Moreover, scaling out the Pods to handle large workload volume while relying on a single shared MySQL database would become a *single point of failure*[11], questions the practicality of such a scheme (to be highly-available and dynamically-scalable).

In order to systematically mitigate the Sybil attacks without degrading the performance of the system, we propose a hybrid design by considering two separate databases. Fig. 4 shows the Memorystore of the VPKIaaS: an in-memory key-value database as a service on GCP, compatible with the Redis [55] protocol, together with a relational database as a service, e.g., MySQL as a service. Each micro-service Pod synchronously interacts with the Redis database to validate a pseudonym request. Upon validating it, the ticket and pseudonyms are issued and the corresponding information is stored in the relational database asynchronously. Such a hybrid design mitigates Sybil attacks without diminishing the overall performance of the pseudonym acquisition process: the time-consuming validation through the relational database is replaced by an efficient validation through the Redis database. Note that MySQL and Redis could both be single point of failures if not offered as a highly-available and dynamically-scalable service. But, a distributed cluster of MySQL is a bottleneck in our scenario because relational databases are slow in nature, especially if the setup is synchronous. The Redis cluster, though, is an in-memory key-value database which offers very fast insertion and query.

3.3.1 *LTCA Sybil Attack Mitigation (Protocol 7)*

The LTCA, the *policy decision and enforcement point* in a domain, issues tickets with non-overlapping intervals, i.e., vehicles cannot obtain tickets with overlapping lifetimes. Upon receiving a ticket request, each LTCA micro-service

11. MySQL as a service can be instantiated to eliminate the single point of failure; however, synchronous interaction with the MySQL would degrade system performance.

Protocol 7 Ticket Request Validation (by LTCA using Redis)

```
1:  procedure ValidateTicketReq($SN^i_{LTC}, tkt^i_{start}, tkt^i_{exp}$)
2:      ($value^i$) ← RedisQuery($SN^i_{LTC}$)
3:      if $value^i == NULL$ OR $value^i <= tkt^i_{start}$ then
4:          RedisUpdate($SN^i_{LTC}, tkt^i_{exp}$)
5:          $Status \leftarrow IssueTicket(.)$          ▷ Invoking ticket issuance procedure
6:          if $Status == False$ then
7:              RedisUpdate($SN^i_{LTC}, value^i$)     ▷ Revert $SN^i_{LTC}$ to $value^i$
8:              return ($False$)                    ▷ Ticket issuance failure
9:          else
10:             return ($True$)                     ▷ Ticket issuance success
11:         end if
12:     else
13:         return ($False$)                        ▷ Suspicious to Sybil attacks
14:     end if
15: end procedure
```

Protocol 8 Pseudonym Request Validation (by PCA using Redis)

```
1:  procedure ValidatePseudonymReq($SN^i_{tkt}$)
2:      ($value^i$) ← RedisQuery($SN^i_{tkt}$)
3:      if $value^i == NULL$ OR $value^i == False$ then
4:          RedisUpdate($SN^i_{tkt}, True$)
5:          $Status \leftarrow IssuePsnyms(.)$      ▷ Invoking pseudonym issuance
6:          if $Status == False$ then
7:              RedisUpdate($SN^i_{tkt}, False$)    ▷ Reverting $SN^i_{tkt}$ to False
8:              return ($False$)                  ▷ Pseudonym issuance failure
9:          else
10:             return ($True$)                   ▷ Pseudonym issuance success
11:         end if
12:     else
13:         return ($False$)                      ▷ Suspicious to Sybil attacks
14:     end if
15: end procedure
```

Pod should check if a ticket was issued to the requester during the requested period. Enforcing such a policy ensures that no vehicle would obtain more than a single valid ticket towards requesting pseudonyms. Thus, given the PCA issues a set of pseudonyms in response to a ticket-enabled request with non-overlapping lifetimes, no vehicle can submit multiple requests for pseudonyms and thus hold at no point in time multiple simultaneously valid pseudonyms. Furthermore, each ticket is implicitly bound to a specific PCA by the vehicle; as a result, the ticket cannot be used more than once to more than one PCAs. Each LTCA micro-service Pod stores the serial number of the vehicle's LTC (as the key) and the expiration time of its current ticket (as the value) on the Redis database. Upon receipt of a new ticket request, each micro-service creates a Redis transaction, a list of commands guaranteed to be executed sequentially without interruption, to validate the ticket (step 7.2).

The Redis transaction checks the existence of the LTC serial number in the database; if it exists, it checks if the request interval overlaps with the previously recorded entry (step 7.3); the request is marked *malicious* if the serial number exists in the database and the requested ticket start time (tkt_{start}) is less than the expiration time of the already issued ticket. Otherwise, the Redis transaction updates the corresponding entry (or adds a new entry if none) with the new ticket expiration time (step 7.4). Then, the procedure for ticket issuance is invoked (step 7.5, i.e., Protocol 3). In case of any failure during the ticket issuance, the ticket expiration value will be rolled back (steps 7.6–7.8). The Redis transaction is executed on a single thread and it guarantees to sequentially execute the commands; thus, even if all replicas of the LTCA received verifiable ticket requests from the same vehicle, Redis ensures that only one ticket request would be served and the rest of them would be denied.

3.3.2 PCA Sybil Attack Mitigation (Protocol 8)

The PCA issues pseudonyms with non-overlapping lifetimes to ensure that no vehicle has more than one valid pseudonym at any point in time. However, to fully eradicate Sybil-based misbehavior, the PCA micro-service also ensures that each ticket to issue a set of pseudonyms for a requester is used only once. Upon receipt of a pseudonym request, each Pod of the PCA micro-service creates a Redis transaction to validate the ticket (step 8.2). If the key (SN_{tkt}) does not exist or the value is false (step 8.3), the procedure for issuing pseudonyms (step 8.5, i.e., Protocol 4) will be invoked (and the appropriate flag is set). In case of failure during the pseudonym acquisition process, the corresponding flag for the ticket will be set to false in the Redis database, i.e., rolling back the transaction, to ensure the consistency of the pseudonym issuance procedure (steps 8.6–8.8). If the flag for the key (SN_{tkt}) is true, the request for obtaining a set of pseudonyms is denied (step 8.13).

3.3.3 DoS-resilient VPKI

In case of a DoS attack, we leverage existing mitigation techniques on the cloud, e.g., [61], [62], [63]. Unlike traditional puzzle mitigation schemes [64], [65], which introduce additional computation or communication overhead, e.g., 50 ms additional overhead to solve a guided tour puzzle [65] in [31], our VPKIaaS benefits from modern strategies to detect and mitigate DoS attacks without such extra overhead. Moreover, the VPKIaaS system can probabilistically verify the batches of CSRs; this renders pseudonym provisioning more efficient. A new process validates each CSR and detects any deviation as misbehavior actions and the RA is informed to initiate a revocation process.

4 Qualitative Analysis

A detailed security and privacy analysis for VPKI entities can be found in [31], [32]. Here, we perform a focused security and privacy analysis concerned with the novel elements and requirements due to the deployment of the VPKIaaS system.

4.1 Security and Privacy Analysis

A deviant LTCA could misbehave by unlawfully registering non-existing vehicles. During the registration process, the LTCA registers a vehicle upon receiving a request from the corresponding OEM. This implies that to fraudulently register a vehicle, two entities must collude. Moreover, the LTCA could try to issue a fake ticket or issue bogus LTCs; but, the RA could perform ticket issuance validation and LTC resolution processes (Protocol 6) to identify a malicious LTCA. Alternatively, a single deviant PCA could issue multiple simultaneously valid pseudonyms for a given vehicle that presents a ticket; or issue pseudonyms without any valid ticket issued by the LTCA. However, through pseudonym validation, the RA requests the corresponding PCA to provide a valid pseudonym commitment key ($CK_{P^i_v}$). Thus, a malicious PCA can be identified if it issued a pseudonym without a valid ticket; and then be evicted from the VPKI system. Note that when performing the pseudonym issuance validation process, the actual identity of the pseudonym owner is not disclosed to the PCA or the RA, i.e., user privacy is preserved. But, when conducting the ticket issuance validation and LTC resolution process, the long-term identity of the vehicle is disclosed to the RA.

Reusing stored tickets to obtain additional pseudonyms by other legitimate vehicles would fail: a ticket is bound to a specific PCA and the PCA keeps records of ticket usage, a ticket cannot be reused for other PCAs. However, a compromised PCA could also bind a pseudonym to a ticket provided by another vehicle when issuing pseudonyms; this implies that the PCA colludes with a malicious vehicle since each vehicle should have verified the validity of commitment keys (CK) for the received pseudonyms set. The RA queries the PCA for the CK_{tkt} and the corresponding ticket. Then, the RA interacts with the LTCA to determine the actual identity of the vehicle. The suspicious vehicle needs to provide the response, obtained during the pseudonyms acquisition process from the PCA. Having the vehicle response, the RA can identify a PCA that misbehaved by binding pseudonyms to a ticket other than the actual one used by the vehicle (and provided by the LTCA). We emphasize that our VPKIaaS scheme does not prevent a malicious PCA from issuing multiple sets of fake pseudonyms; rather, our scheme facilitates efficient identification of a misbehaving PCA by cross-checking the pseudonym issuance procedure in a privacy-preserving manner.

The PCA is evicted from the system and its certificate is revoked. Moreover, all pseudonyms issued by a misbehaving PCA will be revoked. Note that performing this operation could harm user privacy as the actual identity of the vehicle is identified. This operation can be performed under certain circumstances, e.g., if the RA performs pseudonym issuance validation process for two pseudonyms and they both result in the same CK_{tkt}. This clearly is a sign of misbehavior by (at least) one internal VPKI entity.

Sybil-based misbehavior: A malicious vehicle could repeatedly request tickets from the LTCA, and/or aggressively request multiple sets of pseudonyms from the PCA. However, all replicas of a micro-service share a Redis Memorystore to validate every request. Thus, any suspicious request can be *instantaneously* validated through the Redis Memorystore. Redis is executed on a single thread and the transaction is guaranteed to sequentially execute the commands; thus, even if all replicas of a micro-service, e.g., the PCA, received a pseudonym request from one vehicle at the same time, the VPKIaaS system would serve only one pseudonym request and the rest of them would be denied.

The ramification of the Redis service failure[12] depends on the action after the failure, i.e., *fail open* or *fail close*. In case of fail open, Sybil attacks would be possible, as the VPKIaaS system provides vehicles with spurious pseudonyms. Later, it invalidates the erroneously issued credentials by adding them to the CRL. In case of fail close, the VPKIaaS system stops issuing credentials until the failure gets resolved.

DDoS attacks on the VPKIaaS system: Compromised internal entities or external adversaries could try to harm the system operations by launching a DDoS attack, thus degrading the availability of the system. A rate limiting mechanism, by requesting the suspicious nodes to piggyback a response to solve a puzzle [64], [65], prevents internal adversaries from compromising the availability of the system; moreover, the system flags misbehaving users, thus evicting them from the system. External adversaries, or internal ones that do not use their credentials, could launch a DDoS attack by clogging the LTCA with a fake request for a ticket using a bogus signature or a bogus certificate, or the PCA with bogus tickets. In fact, such misbehaving (external) entities attempt to compromise the availability of the VPKI entities by mandating them to excessively validate the signature of fake LTCs or bogus tickets [37], [31], i.e., performing a signature flooding attack [38].

Compromised *internal* vehicles (with valid tickets) could conduct a DDoS attack on the PCA during the pseudonym acquisition process: each malicious vehicle would construct legitimate-looking CSRs with bogus proof-of-possession signature for the last one. This would impose significant computation overhead on the PCA, requiring validation of all the CSRs; the most time-consuming part of pseudonym validation request. To prevent internal malicious vehicles from clogging a DDoS attack on the PCA, we modified the protocol (as presented in [31], [45]): for each CSR, each vehicle needs to sign the hash of concatenated *all* pseudonymous public keys with each private key. Thus, with each CSR, a vehicle would non-repudiably attest to the possession of the private keys corresponding to all the public keys. This does not impose extra overhead on the performance as the signature generation and validation processes remain intact; the difference is that the hash of multiple public keys per CSR is signed and verified instead of signing and verifying the hash of a single public key.

This brings forth the advantage of probabilistic selection of the CSRs for verification, e.g., during a rush hour scenario or when under a DDoS attack, without compromising the security of the scheme or imposing computation overhead. Note that detecting a malformed CSR cannot result in a false positive. In case of the probabilistic selection of the CSRs for validation, a new process will validate each CSR during system *idle* time[13]. If any of proof-of-possession validation fails, this could be identified as a misbehavior action and be reported to the RA. Later, misbehaving vehicles would be (temporarily) evicted from the system and revocation information (CRL/Δ-CRL [56]) would be efficiently disseminated across the network, e.g., [28], [32].

We achieve high-availability and fault-tolerance in the face of a benign failure by exploiting the Kubernetes master to kill the running (faulty) Pod, e.g., in case of system faults or crashes, and create a new Pod. In case of resource depletion attacks, the Kubernetes master scales out the Pods to handle such demanding loads. At the same time, a puzzle technique, e.g., [64], [65], can be employed as a mitigation approach in a VPKI [31]: before initiating the pseudonym acquisition process, each vehicle is mandated to visit a predefined set of nodes, in a pre-determined sequential order

12. A failure could be benign, e.g., system faults or crashes, or under resource depletion attacks, e.g., a DDoS attack.

13. Pseudonyms are issued with non-overlapping lifetimes; thus, an internal malicious vehicle can only use one pseudonym at any time. This would enable timely validation of successive CSRs by the PCA.

TABLE 2: VPKI Secrecy Analysis for Dolev-Yao Adversaries.

User-sensitive Piece of Information	Entity	Secrecy	Strong Secrecy (Unlinkability)
Vehicle Id (LTC)	V, LTCA	✓	✓
Ticket ($tkt/n\text{-}tkt$)	V, LTCA, PCA	✓	✓
Pseudonym CSR ($(K_v^1, \dots, K_v^n)_{\sigma_{k_v^i}}$)	V, PCA	✓	✓
Pseudonym ($(P_v^i)_{\sigma_{pca}}$)	V, LTCA, PCA	✓	✓
Timestamps (t_s, t_e)	V, LTCA, PCA	✓	✓
Random number (Rnd_{tkt})	V, LTCA	✓	✓
Random number ($Rnd_{CK_{tkt}}$)	V, LTCA, PCA	✓	✓
Random number (Rnd_{psnym})	V, PCA	✓	✓
Ticket Commitment Key (CK_{tkt})	V, LTCA, PCA	✓	✓
Pseudonym Commitment Key (CK_P)	V, PCA	✓	✓

to solve a puzzle [65]. The system grants access once the requester visits the pre-defined set of nodes as required. As a result, the computation power of an attacker is degraded to the computation power of a legitimate client [65], thus, an adversary cannot send high-rate spurious requests to the VPKI. On the side of the infrastructure, there are DDoS mitigation techniques at different network layers, provided by various cloud service providers[14], e.g., [61], [62], [63].

Synchronization among the VPKI entities: Lack of synchronization between the LTCA and the PCA could affect the pseudonym issuance process, e.g., a PCA would not issue pseudonyms for a seemingly 'expired' ticket. However, mildly drifting clocks of the VPKI entities can hardly affect the operation, because the pseudonym lifetimes and periods for pseudonym refills (Γ) are in the order of minutes, typically. It suffices VPKI entities periodically synchronizing their clocks. For example, if the accuracy of a Real Time Clock (RTC) is 50 parts-per-million (ppm), i.e., 50×10^{-6}, and the maximum accepted error in timestamp is 50 ms, then each entity should synchronize its clock every 16 minutes.

4.2 Formal Analysis

We formalize the security of the ticket and pseudonym acquisition protocols, i.e., the principal processes of the VPKI system, using ProVerif [68], [69]: an automated protocol verifier which enables the modeling of security protocols in π Calculus [70]. This would increase confidence in the analysis of the security protocols. System entities (VPKI) and clients (requesting vehicles) are modeled as processes and protocols are represented as parallel compositions of multiple processes. Each VPKI entity possesses its own private keys and cryptographic credentials, and the adversaries does not have access (knowledge) on the private keys of legitimate entities. ProVerif verifies security protocols in the presence of Dolev-Yao adversaries [71]: with complete control of the communication channel, they eavesdrop, modify, delete, and forge any message using strictly cryptographic keys they possess. As part of the future work, we plan to formally analyze pseudonym issuance validation (Protocol 5) and pseudonym resolution (Protocol 6) processes.

Table 2 summarizes our findings: VPKIaaS system ensures *secrecy* and *strong secrecy* for all critical pieces of information during ticket and pseudonym acquisition protocols; the results ensure system security and user privacy protection. Secrecy refers to the fact that an attacker does not obtain (or infer) the value of a secret value itself [70]. However, strong secrecy implies that an attacker cannot see differences when the value of a secret changes [70]. To examine if strong secrecy holds for a given datum, we use *noninterf*. Based on the ProVerif outputs, our security protocols guarantee secrecy and strong secrecy; thus, adversaries cannot infer changes over the exchanged data. For example, given two distinct tickets, belonging to the same vehicle, it is infeasible for an adversary to relate the two [72]; the same holds for other user sensitive data, e.g., LTC, pseudonyms, and timestamps. In the Dolev-Yao adversarial model [71], unlinkability [73], [74] is achieved[15].

5 Quantitative Analysis

Experimental setup: We leveraged the state-of-the-art VPKI system [31] and restructured its source code to fit in a microservices architecture through containerization, automation, and bootstrapping of services. We further added health and load metric publishing features, to be used by an orchestration service to scale in/out accordingly. We built and pushed Docker images for LTCA, PCA, RA, MySQL, and Locust [75], *an open source load testing tool*, to the Google Container Registry [76]. Isolated namespaces and deployment configuration files are defined before Google Kubernetes Engine (GKE) v1.10.11 [77] cluster runs the workload. We configured a cluster of five Virtual Machines (VMs) (n1-highcpu-32), each with 32 vCPUs and 28.8GB of memory. The implementation is in C++ and we use FastCGI [78] to interface Apache web-server. We use XML-RPC [79] to execute a remote procedure call on the cloud. The VPKIaaS interface is language-neutral and platform-neutral, as we use Protocol Buffers [80] for serializing and deserializing structured data[16]. For the cryptographic protocols and primitives (ECDSA and TLS), we use OpenSSL with ECDSA-256 key pairs according to the ETSI (TR-102-638) [3] and IEEE 1609.2 [2] standards; other algorithms and key sizes are compatible in our implementation.

To facilitate the deployment of the VPKIaaS, we created all VPKIaaS configuration in YAML language [82], applicable to deploy on any cloud provider which offers *Kubernetes as a Service*, e.g., GCP [53] and Amazon Web Service (AWS) [83]. For our experiments, we deployed our VPKIaaS on the GKE [77]. We also used other GCP services: *Memorystore* [84], *Prometheus* [85], and *Grafana* [86]. The Memorystore service is a Redis-compatible [55] service (16GB, Version 6.x, Redis Standard with a failover replica in a separate zone for high availability [87]) which acts as in-memory key-value data store (see Fig. 4). Prometheus is a feature-rich metric service which collects all the metrics of the Kubernetes cluster and the applications running on it into a time-series database. We use Grafana to visualize the metrics collected by Prometheus and monitor the *system under test*. Prometheus and Grafana are deployed as prepared applications from the GCP marketplace [88] on the Kubernetes cluster. Moreover,

14. There exist DoS attacks on the TLS/SSL establishment, e.g., SYN (synchronize) floods and SSL Garbage Flood [66], [67]. All these can be mitigated by leveraging the existing DoS countermeasures on the cloud, e.g., [61], [62], [63]. Further discussion is orthogonal to this investigation.

15. We provided the Proverif code in the complementary document.

16. One can interact with the VPKIaaS system using *SecProtoBuf* [81] to automate the signature generation and validation procedures.

TABLE 3: Experiment Parameters.

Parameters	Config-1	Config-2	Config-3
total number of vehicles	1000	100, 50,000	5000
hatch rate	1	1, 100	5, 10, 15, 20, 25
interval between requests	1000-5000 ms	1000-5000 ms	30000-60000 ms
pseudonyms per request	100, 200, 300, 400, 500	100, 200, 500	100, 200
LTCA memory request	128 MiB	128 MiB	128 MiB
LTCA memory limit	256 MiB	256 MiB	256 MiB
LTCA CPU request	500 m	500 m	500 m
LTCA CPU limit	1000 m	1000 m	1000 m
LTCA HPA	1-40; CPU 60%	1-40; CPU 60%	1-40; CPU 60%
PCA memory request	128 MiB	128 MiB	128 MiB
PCA memory limit	256 MiB	256 MiB	256 MiB
PCA CPU request	700 m	700 m	700 m
PCA CPU limit	1000 m	1000 m	1000 m
PCA HPA	1-120; CPU 60%	1-120; CPU 60%	1-120; CPU 60%

we leveraged Locust [75], deployed on the Kubernetes cluster, to synthetically generate a large volume of requests.

Fully-automated Certificate Management on the GCP: Leveraging scripting languages, we created tools to facilitate automated compilation, integration, and deployment of the micro-services on the cloud. Having finished the compilation and installation scripts, Docker images for each micro-service are ready to push on the Google Container Registry [76].

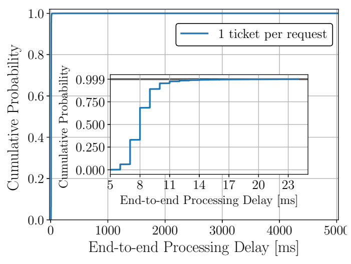
Metrics: To evaluate the performance of our VPKIaaS system, we measure the latency to obtain pseudonyms under different scenarios and configurations for a large-scale mobile environment. More specifically, we evaluate the performance of the system with (and without) flash crowds to illustrate its *high-availability*, *robustness*, *reliability*, and *dynamic-scalability*. We demonstrate the performance of our VPKIaaS system by emulating a large volume of synthetic workload. Table 3 shows the configurations used in our experiments, with *Config-1* referring to a *'normal'* vehicle arrival rate and *Config-2* and *Config-3* for a *flash crowd* scenario. Experiments with *Config-1* indicates that every 1-5 seconds, a new vehicle joins the system and requests a batch of 100-500 pseudonyms. To emulate a flash crowd scenario, i.e., *Config-2* and *Config-3*, beyond having vehicles joining the system based on *Config-1*, new vehicles (with hatch rates[17] 5, 10, 15, 20, 25, and 100) join the system every 1-5 seconds and request a batch of 100, 200, and 500 pseudonyms.

Remark: Assuming the pseudonyms are issued with non-overlapping intervals (important to mitigate Sybil-based misbehavior), obtaining 100 and 500 pseudonyms per day implies pseudonyms lifetimes of 14.4 minutes or 172.8 seconds, respectively. According to actual large-scale urban vehicular mobility dataset, e.g., Tapas-Cologne [89] or LuST [90], the average trip duration is 10 to 30 minutes. According to the US DoT, the average daily commute time in the US is around 1 hour [21]. Thus, requesting 500 pseudonyms per day would cover 24 hours trip duration with each pseudonym lifetime of approx. 3 minutes. We evaluate the performance of our VPKIaaS system under such seemingly extreme configurations (Config-1 and Config-2 in Table 3).

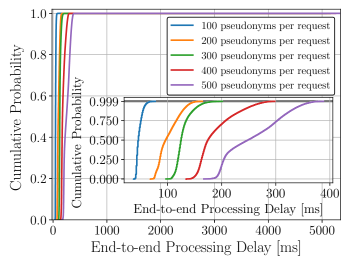
5.1 Large-scale Pseudonym Acquisition

Fig. 5.a illustrates the Cumulative Distribution Function (CDF) of the single ticket issuance processing delay (executed based on Config-1 in Table 3); as illustrated, 99.9% of ticket

17. The hatch rate is the number of vehicles to spawn per second [75].

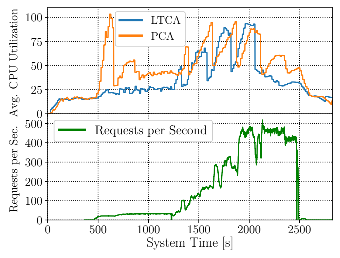


(a) Ticket Issuance

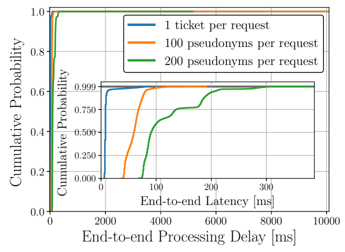


(b) Pseudonyms Issuance

Fig. 5: (a) CDF of end-to-end latency to issue a ticket. (b) CDF of end-to-end processing delay to issue pseudonyms.



(a)



(b)

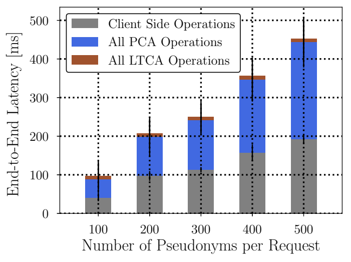
Fig. 6: VPKIaaS system in a flash crowd load situation. (a) CPU utilization and the number of requests per second. (b) CDF of processing latency to issue tickets and pseudonyms.

requests are served within 24 ms: $F_x(t = 24\,ms) = 0.999$, i.e., $Pr\{t \leq 24\,ms\} = 0.999$. Fig. 5.b shows the CDF of processing latency for issuing pseudonyms with different batches of pseudonyms per request as a parameter. For example, with a batch of 100 pseudonyms per request, 99.9% of the vehicles are served within less than 77 ms: $F_x(t = 77\,ms) = 0.999$. Even with a batch of 500 pseudonyms per request, the VPKIaaS system can efficiently issue pseudonyms: $F_x(t = 388\,ms) = 0.999$. The results confirm that the VPKIaaS scheme is efficient and scalable: the pseudonym acquisition process incurs low latency and it efficiently issues pseudonyms for the requesters.

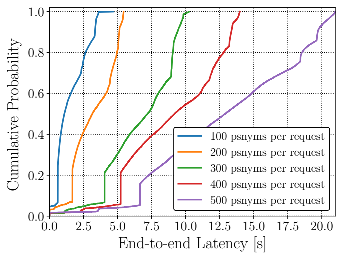
5.2 VPKIaaS with Flash Crowd Load Pattern

Fig. 6 shows the performance of the VPKIaaS during a surge in pseudonym acquisition requests (based on Config-2 in Table 3, with 100 pseudonyms per request). We assess the CPU utilization of the LTCA and the PCA Pods (Fig. 6.a top) and the total number of pseudonyms requests per second (Fig. 6.a bottom). When the number of requests per second increases, the average CPU utilization rises; however, when CPU utilization hits a 60% threshold, defined in the Horizontal Pod Autoscalers (HPAs) [91], the LTCA and the PCA deployment horizontally scales out to handle the loads, thus the average CPU utilization drops.

Fig. 6.b shows the end-to-end processing latency to obtain tickets and a batch of 100 or 200 pseudonyms in a flash crowd situation. The processing latency to issue a single ticket is: $F_x(t = 87\,ms) = 0.999$; to issue a batch of 100 pseudonyms per request, the processing latency is: $F_x(t = 192\,ms) = 0.999$. Compared to the end-to-end processing delay in 'normal' conditions (Fig. 5), the processing latency of issuing a single ticket increases from 24 ms to 87 ms;

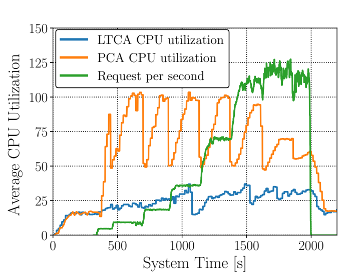


(a)

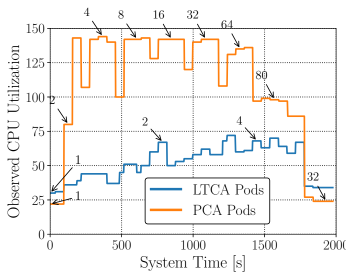


(b)

Fig. 7: VPKIaaS system with flash crowd load pattern. (a) Average end-to-end latency to obtain pseudonyms. (b) CDF of end-to-end latency, observed by clients.



(a)

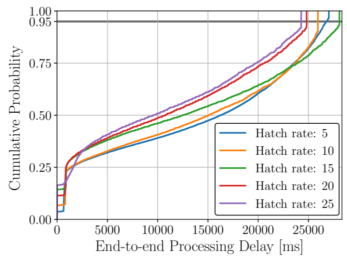


(b)

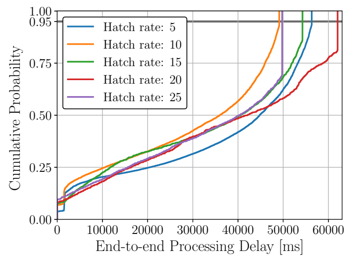
Fig. 8: Each vehicle requests 500 pseudonyms (CPU utilization observed by HPA). (a) Number of active vehicles and CPU utilization. (b) Dynamic scalability of VPKIaaS system.

the processing latency to issue a batch of 100 pseudonyms increases from 77 ms to 192 ms. Thus, even with such a highly demanding request rate, the VPKIaaS system issues credentials efficiently. Throughout the simulated scenario, the total number of vehicles requesting 100 pseudonyms (under Config-2 in Table 3) is 398870 and the VPKIaaS system issues approximately 40 millions pseudonyms within 2500 seconds; we can compute that at the rate of this scenario, the VPKIaaS system would issue 0.5×10^{12} pseudonyms over a year. This number is smaller than the expected number of pseudonyms to be issued per year, i.e., 1.5×10^{12} [21] (see Sec. 1). Note that this is a proof of concept of the VPKIaaS system implementation; by allocating more resources and increasing the pseudonym request rates, the VPKIaaS system would issue more numerous pseudonyms.

Fig. 7.a shows the latency for each system component to obtain different size batches of pseudonyms per request (Config-2 in Table 3). Our VPKIaaS system outperforms prior work [47]: the processing delay to issue 100 pseudonym for [47] is approx. 2000 ms, while it is approx. 56 ms in our system, i.e., achieving a 36-fold improvement over prior cloud-based work [47]. Fig. 7.b illustrates the average end-to-end latency to obtain pseudonyms, observed by clients: during a surge of requests, *all* vehicles obtained a batch of 100 pseudonyms within less than 4900 ms (including the networking latency). Obviously, the shorter the pseudonym lifetime, the higher the workload on the VPKI, thus the higher the end-to-end latency. Note that serving requests in a flash crowd scenario at this rate (Config-2 in Table 3) implies serving 720,000 vehicles joining the system within an hour. Thus, even under such flash crowd load pattern, our



(a) 100 Pseudonyms per Request



(b) 200 Pseudonyms per Request

Fig. 9: Pseudonym acquisition with SECMACE [31].

VPKIaaS system comfortably handles the high demand.

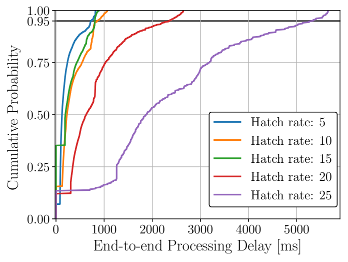
5.3 Dynamic-scalability of the VPKIaaS

In this scenario, we evaluate the reliability and dynamic scalability of our VPKIaaS system. To emulate a large volume of workload, we generated synthetic workload using 30 containers, each with 1 vCPU and 1GB of memory (executed based on Config-2 in Table 3). Fig. 8.a shows the average CPU utilization of the LTCA and the PCA Pods (observed by HPA), as well as the total number of requests per second. Fig. 8.b shows how our VPKIaaS system dynamically scales out or scales in, according to the rate of pseudonyms requests. The numbers next to the arrows show the number of LTCA and PCA Pod replicas at any specific system time. As illustrated, the number of PCA Pods starts from 1 and it gradually increases; at system time 1500, there is a surge in pseudonym requests, thus the number of PCA Pods increases to 80. Note that issuing a ticket is more efficient than issuing pseudonyms; thus, the LTCA micro-service scaled out only up to 4 Pod replicas.

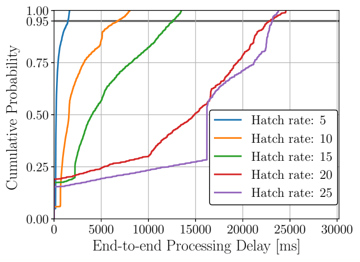
5.4 VPKIaaS Performance Comparison

We compare VPKIaaS scheme with a *baseline* scheme [47], which implements a VPKI according to the ETSI architecture [3]. More precisely, each vehicle requests pseudonyms from an authorization authority [3], [47]; the request is forwarded to the enrollment authority to check and validate the request. Upon a successful validation, the authorization authority issues the pseudonyms and sends them back to the vehicle. Using the similar setup to have a meaningful and direct comparison, we achieve a 36-fold improvement over the baseline scheme: under normal conditions, the processing delay to issue 100 pseudonyms for the baseline scheme is approx. 2000 ms, while it is approx. 56 ms in the VPKIaaS system. Even under a flash crowd scenario (based on Config-2), the processing delay to issue 100 pseudonyms is approx. 71 ms, i.e., 28-fold improvement. Furthermore, unlike the VPKI system in [47], our implementation supports dynamic scalability, i.e., the VPKI scales out, or scales is, based on the arrival rate of pseudonyms requests.

Fig. 9 and Fig. 10 compare the performance of the SECMACE [31] and the VPKIaaS systems under Config-3. Vehicles join the system at different hatch rates and request for 100 and 200 pseudonyms. At a hatch rate of 5, the number of new vehicles, joining the system per second, is five. With the SECMACE system (with a hatch rate of 25), 99.9% of 100 pseudonyms requests are served within 30 seconds:



(a) 100 Pseudonyms per Request



(b) 200 Pseudonyms per Request

Fig. 10: Pseudonym acquisition with SECMACE+ (VPKIaaS).

$F_x(t = 30\ sec) = 0.999$ (Fig. 9.a). However, with the same set up, the VPKIaaS system can serve requesters within 5.5 seconds: $F_x(t = 5.5\ sec) = 0.999$ (Fig. 10.a). Similarly, the VPKIaaS system outperforms the SECMACE system when requesting 200 pseudonyms: for SECMACE with hatch rate of 10, $F_x(t = 50\ sec) = 0.999$ while for the VPKIaaS system, $F_x(t = 15\ sec) = 0.999$.

Fig. 11 compares the performance of SECMACE [31] and the VPKIaaS in flash crowd conditions. The SECMACE performance drastically decreases when there is a surge in the pseudonym arrivals; on the contrary, the VPKIaaS system can comfortably handle such demanding requests while efficiently issuing batches of pseudonyms. To issue 200 pseudonyms with a hatch rate of 10, it takes 29273 ms for the SECMACE while it takes 2294 ms for the SECMACE+, i.e., 13 times more efficient in issuing pseudonyms.

6 Related Work

A VPKI can provide vehicles with valid pseudonyms for a long period, e.g., 25 years [30]. However, extensive preloading with millions of pseudonyms per vehicle for such a long period is computationally costly, inefficient in terms of utilization and cumbersome for revocation [29], [32]. On the contrary, several proposals suggest more frequent Vehicle-to-VPKI interactions, namely *on-demand* schemes, e.g., [37], [31], [92], [93]. This strategy provides more efficient pseudonym utilization and revocation, thus being effective in fending off misbehavior. However, for on-demand pseudonym acquisition, one needs to design (and deploy) an efficient and scalable system while being resilient against any resource depletion attack. Even though VPKI systems may handle large-scaled distributed scenarios, e.g., [47], there is lack of dynamic scalability (i.e., dynamically scale out/in according to the arrival rates) and resilient to a resource depletion attack, e.g., a DoS attack. Beyond a significant performance improvement over [47], our VPKIaaS implementation is highly-available, dynamically-scalable, and fault-tolerant.

Sybil-based [42] misbehavior can seriously affect the operation of VC systems, as multiple fabricated non-existing vehicles could pollute the network by injecting false information. For example, an adversary with multiple valid pseudonyms, termed here a *Sybil* node, could create an illusion of traffic congestion towards affecting the operation of a traffic monitoring system, or broadcast fake misbehavior detection votes [94], [95], [96], or disseminate Spam to other users in a vehicular social network [24]. The idea of enforcing

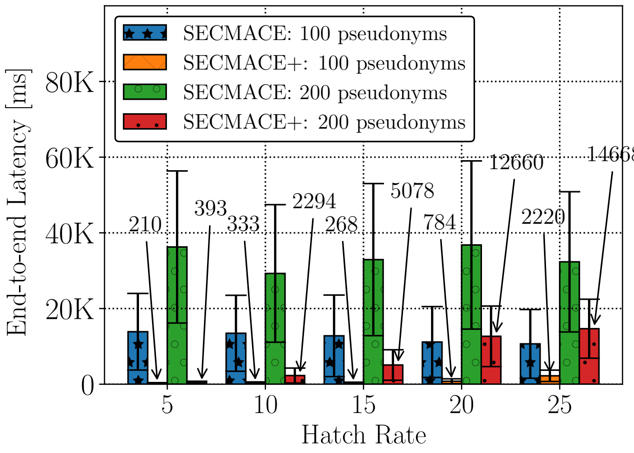


Fig. 11: Pseudonym acquisition comparison between SECMACE [31] and SECMACE+ (VPKIaaS with Redis Enabled).

non-overlapping pseudonym lifetimes was first proposed in [10]. This prevents an adversary from equipping itself with multiple valid identities, and thus affecting protocols of collection of multiple inputs, e.g., based on voting, by sending out redundant false, yet authenticated, information. Even though this idea has been accepted, a number of proposals, e.g., [14], [30], do not prevent a vehicle from obtaining simultaneously valid pseudonyms via multiple pseudonym requests. The existence of multiple pseudonym issuers deteriorate the situation: a vehicle could request pseudonyms from multiple service providers, while each of them is not aware whether pseudonyms for the same period were issued by any other service provider. One can mitigate this vulnerability by relying on an HSM, ensuring all signatures are generated with a single private key corresponding to a single valid pseudonym at any time. There are also distributed schemes to detect Sybil nodes based on radio characteristics and triangulation [43], [44]. Such strategies are application-dependent: this cannot guarantee the operation of a traffic monitoring system from an adversary who disseminates multiple traffic congestion messages, each signed under a distinct 'fake' private key.

V-tokens scheme [93] prevents a vehicle from obtaining multiple simultaneously valid pseudonyms due to having service providers communicating with each other, e.g., a distributed hash table. SECMACE [31] (including its predecessors [37], [41]) prevents Sybil-based misbehavior on the infrastructure side without the need for an additional entity, i.e., extra interactions or intra-VPKI communications. More specifically, it ensures each vehicle has one valid pseudonym at any time in a multi-domain environment. However, when deploying SECMACE (designed for and implemented on a single machine) on the cloud, a malicious vehicle could repeatedly request pseudonyms, hoping that requests are delivered to different replicas of a micro-service, thus obtaining multiple simultaneously valid pseudonyms, e.g., [46], [47]. Unlike such schemes, our VPKIaaS scheme prevents Sybil-based misbehavior on the cloud-deployed infrastructure: it ensures that each vehicle can only have one valid pseudonym at any time without affecting the timely issuance of pseudonyms.

Moreover, to handle a large volume workload, SECMACE [31] requires to statically allocate resources. In case of an unpredictable surge in the arrival rates or being under a DDoS attack, the performance of SECMACE would drastically decrease. Also, when deploying SECMACE on the cloud, a malicious vehicle could repeatedly request to obtain pseudonyms towards performing Sybil-based misbehavior. On the contrary, our VPKIaaS system can comfortably handle requests with unexpected arrival rate while being *efficient* in issuing pseudonyms, being *resilient* against Sybil and resource depletion attacks, and being cost-effective by systematically allocating and deallocating resources.

The VPKI entities are, often implicitly, assumed to be fully trustworthy. Given the experience from recent mobile applications, e.g., [97], [98], [99], the adversarial model is extended from fully trustworthy to *honest-but-curious* VPKI servers, notably in [14], [31]. Such honest-but-curious entities may subvert the security protocols and deviate from system policies if gained an advantage without being identified, e.g., inferring user sensitive information [100], [101], [102], [103], [104], [105]. During the pseudonym acquisition process, a PCA could trivially link all pseudonyms for a requester issued within an interval (Γ). However, one can configure the VPKI system to issue fully-unlinkable pseudonyms [31]: each vehicle requests a distinct ticket to obtain each pseudonym. Thus, even an honest-but-curious PCA cannot link two successive pseudonym requests to a single vehicle. During the pseudonym issuance process, the PCA calculates the *Revocation Commitment Key* [28], [32]. This essentially provides two important features: firstly, it prevents a compromised PCA from mapping a different ticket during the resolution process. Secondly, this enables efficient distribution of the CRL: the PCA implicitly correlates a batch of pseudonyms belonging to each requester. Thus, upon revocation, the PCA only needs to include one entry per batch of pseudonyms without compromising their unlinkability.

In case of fully-unlinkable pseudonyms, there is only one pseudonym issued per interval (Γ); upon revocation, the PCA discloses one entry per Γ. This guarantees that even after a revocation event, the PCA cannot link any two pseudonyms belonging to the same vehicle [28], [32] (unlike the SCMS design [14]). In the SCMS design [14], this linking attack is mitigated by a Registration Authority, one PCA, and two Linkage Authorities (LAs), i.e., four entities. In contrast, our scheme achieves the same with a simpler design, based on two VPKI entities, the LTCA and the PCA.

In general, SECMACE [31], [28], [45] issues numerous pseudonyms per Γ, with each pseudonym having a lifetime $\tau_p \ll \Gamma$. For SCMS [14], the interval (Γ) is equal to pseudonym lifetime (τ_p), by default, and multiple pseudonyms (20-40) are issued within an interval Γ. Thus, upon revocation, all 20-40 pseudonyms could be linked within a Γ. SECMACE assumes a shorter Γ, e.g., 1 hour, with a pseudonym lifetime of 5 minutes. Upon revocation, one cannot link the pseudonyms backward due to the utilization of a hash chain. The shorter the interval is, the narrower the linkability window becomes. But simply narrowing down this interval does not diminish the linkability window in ACPC [33] and SCMS [14], [30]: for each vehicle, 20-40 pseudonyms are issued, valid for a week, all with overlapping validity intervals.

To diminish linkability, the pseudonym issuance process should be changed, i.e., issuing pseudonyms with a shorter validity interval. More precisely, if the pseudonyms are issued with a validity interval of one week, defining a shorter interval for CRL distribution becomes meaningless because the revoked pseudonyms would appear in all shorter-defined intervals. The second important issue is that when the linkage seed is disclosed, all the revoked pseudonyms, within that interval, will be linked; thus, if any of such revoked pseudonyms have been used prior to revocation, they become linkable, thus, harming user privacy. On the contrary, our scheme mitigates this vulnerability based on what we term Pseudonym Acquisition Policies [31], [41]: vehicles request one or multiple pseudonyms based on a predetermined universally fixed interval and the pseudonym lifetime. This prevents linking pseudonyms based on their timing information, linking multiple sets of pseudonyms belonging to the same requester, and most important, facilitating more efficient dissemination of revocation information.

SEROSA [106] proposed a general service-oriented security architecture seeking to bridge Internet and the VC domains. The LTCA issues authenticated, yet anonymized, tickets to the vehicles to obtain pseudonyms from the PCA. However, the LTCA can learn from pseudonym acquisition process: when and from which PCA the vehicle will obtain pseudonyms since the Security Assertion Markup Language (SAML) token is presented to the LTCA. The exact pseudonym acquisition period could be used to infer the active period of the vehicle operation, and the targeting PCA could be used to infer the approximate location (assuming the vehicle chooses the nearest PCA) or the affiliation (assuming the vehicle can only obtain pseudonyms from the PCA in the domain it is affiliated to, or operating in) of the vehicle. Moreover, the multi-domain environment explicitly addressed by [106] leaves space for Sybil-based misbehavior: the infrastructure cannot prevent multiple spurious requests to different PCAs. Furthermore, our scheme outperforms SEROSA [106] with a 2.5-fold improvement [31]. The main reasons for such a significant improvement is efficient protocol design with minimum interaction with the VPKI (only two interactions rather than three interactions in [106] and [47]) as well as leveraging multiple programming features, e.g., multi-threading implementation, code and memory usage optimization techniques.

The trend towards the cloudification of the PKIs is not limited to the VC systems; there are various cloud-based PKIs for the Internet, e.g., Enterprise Java Beans Certificate Authority (EJBCA) [107], the Key-factor PKIaaS [108], the digicert [109], and the HydrantID [110]. Such PKI as a Service (PKIaaS) systems cannot be used for the VC systems. To comply with the security and privacy requirements in the standardization bodies (IEEE 1609.2 [2] and ETSI [3], [4]), no single entity should be able to fully de-anonymize a user or link successive anonymized credentials over a long period of time. This requires a special-purpose PKI with *separation of duties* to ensure *conditional anonymity* for the deployment of secure and privacy-protecting VC systems.

Outside the VC realm, there are also different proposals for PKIs to be resilient against *compromised* insiders. Such schemes rely on signing a certificate by more than a threshold number of CAs, e.g., [111], [112]; however, such schemes cannot be used by VC systems. For example, issuing a certificate in [112] takes approximately 2 minutes and it varies with the number of required CAs. Obviously, this contradicts with *on-demand* pseudonym acquisition strategies for VC systems, e.g., [31], [113], [40], [41], which necessitate efficient pseudonym provisioning.

7 Conclusion

The deployment of secure and privacy-preserving VC systems relies on deploying a special-purpose Vehicular Public Key Infrastructure (VPKI). Its success requires viability in terms of performance and cost. To achieve high availability, resiliency, scalability, and cost-effective VPKI deployment, we leverage the state-of-the-art VPKI, enhance its functionality, and migrate it into the GCP. Through extensive security and privacy analysis, we show that the VPKIaaS system fully eradicates Sybil-based misbehavior without compromising the efficiency of the pseudonym acquisition process. We formally analyze the security protocols for ticket and pseudonym acquisition processes: VPKIaaS achieves security, privacy and resilience in the presence of strong adversaries. Moreover, we improve the system to be resilient against deviant LTCA and RA entities that aim to deteriorate the security of the system or harm user privacy. With these characteristics, our VPKIaaS system can catalyze the deployment of secure and privacy-preserving VC systems.

Acknowledgement

Work supported by the Swedish Foundation for Strategic Research (SSF), the KAW Academy Fellows program (WAF), and the Swedish Science Foundation (VR).

References

[1] "Review of NHTSA Proposal to Mandate V2V Communication for Safety," https://www.etsi.org/deliver/etsi_ts/102700_102799/102731/01.01.01_60/ts_102731v010101p.pdf, Tech. Rep., Dec. 2016.

[2] "IEEE Standard for Wireless Access in Vehicular Environments - Security Services for Applications and Management Messages," Tech. Rep., Mar. 2016.

[3] "Intelligent Transport Systems (ITS); Vehicular Communications; Basic Set of Applications; Definitions," https://www.etsi.org/deliver/etsi_tr/102600_102699/102638/01.01.01_60/tr_102638v010101p.pdf, Tech. Rep., Jun. 2009.

[4] "Intelligent Transport Systems (ITS); Vehicular Communications; Basic Set of Applications; Part 2: Specification of Cooperative Awareness Basic Service - V1.4.1," Tech. Rep., Apr. 2019.

[5] "Intelligent Transport Systems (ITS); Security; Security Header and Certificate Formats," Tech. Rep., Oct. 2017.

[6] "C-ITS Platform Phase II: Certificate Policy for Deployment and Operation of European Cooperative Intelligent Transport Systems (C-ITS)," https://ec.europa.eu/transport/sites/default/files/c-its_certificate_policy_release_1.pdf, Tech. Rep., Jun. 2017.

[7] PKI-Memo, "C2C-CC," http://www.car-2-car.org/, Tech. Rep., Feb. 2011.

[8] P. Papadimitratos, L. Buttyan, T. Holczer, E. Schoch, J. Freudiger, M. Raya, Z. Ma, F. Kargl, A. Kung, and J.-P. Hubaux, "Secure Vehicular Communication Systems: Design and Architecture," *IEEE Communications Magazine*, vol. 46, no. 11, Nov. 2008.

[9] A. Kung, "Security Architecture and Mechanisms for V2V/V2I, SeVeCom," https://sevecom.eu/Deliverables/Sevecom_Deliverable_D2.1_v3.0.pdf, Tech. Rep., Feb. 2008.

[10] P. Papadimitratos, L. Buttyan, J.-P. Hubaux, F. Kargl, A. Kung, and M. Raya, "Architecture for Secure and Private Vehicular Communications," in *International Conference on ITS Telecommunications (IEEE ITST)*, Sophia Antipolis, Jun. 2007, pp. 1--6.

[11] PRESERVE-Project, www.preserve-project.eu/, Jun. 2015.

[12] "Vehicle Safety Communications (VSC)," https://www.campllc.org/vehicle-safety-communications-vsc/, Tech. Rep., Dec. 2006.

[13] "Vehicle Safety Communications - Applications (VSC-A) - Final Report," https://www.nhtsa.gov/sites/nhtsa.gov/files/811492a.pdf, Tech. Rep., Sep. 2011.

[14] W. Whyte, A. Weimerskirch, V. Kumar, and T. Hehn, "A Security Credential Management System for V2V Communications," in *IEEE Vehicular Networking Conference*, Boston, MA, Dec. 2013.

[15] "Vehicle Safety Communications Security Studies: Technical Design of the Security Credential Management System," https://www.regulations.gov/document?D=NHTSA-2015-0060-0004, Jul. 2016.

[16] "Development of DSRC Device and Communication System Performance Measures - Recommendations for DSRC OBE Performance and Security Requirements," https://rosap.ntl.bts.gov/view/dot/31627, Tech. Rep., May 2016.

[17] "Intelligent Transport Systems (ITS); Security; Pre-standardization Study on Pseudonym Change Management," https://www.etsi.org/deliver/etsi_tr/103400_103499/103415/01.01.01_60/tr_103415v010101p.pdf, Tech. Rep., Apr. 2018.

[18] "Let's Encrypt Stats," https://letsencrypt.org/, Oct. 2018.

[19] "Comodo Certification Authority," ssl.comodo.com/, Oct. 2018.

[20] "Symantec SSL/TLS Certificates," https://www.websecurity.digicert.com/ssl-certificate, Oct. 2018.

[21] "V2V Communications: Readiness of V2V Technology for Application," Tech. Rep., Aug. 2014, National Highway Traffic Safety Administration, DOT HS 812 014.

[22] P. Papadimitratos, A. La Fortelle, K. Evenssen, R. Brignolo, and S. Cosenza, "Vehicular Communication Systems: Enabling Technologies, Applications, and Future Outlook on Intelligent Transportation," *IEEE Communications Magazine*, vol. 47, no. 11, pp. 84--95, Nov. 2009.

[23] R. Shokri, G. Theodorakopoulos, P. Papadimitratos, E. Kazemi, and J.-P. Hubaux, "Hiding in the Mobile Crowd: Location Privacy through Collaboration," *IEEE Transactions on Dependable and Secure Computing (TDSC)*, vol. 11, no. 3, pp. 266--279, May 2014.

[24] H. Jin, M. Khodaei, and P. Papadimitratos, "Security and Privacy in Vehicular Social Networks," in *Vehicular Social Networks*. Taylor & Francis Group, Mar. 2016.

[25] E. Topalovic, B. Saeta, L.-S. Huang, C. Jackson, and D. Boneh, "Towards Short-lived Certificates," *IEEE Oakland Web 2.0 Security and Privacy (W2SP)*, May 2012.

[26] P. McDaniel and A. Rubin, "A Response to "Can We Eliminate Certificate Revocation Lists?"," in *FC (Springer)*, Berlin, Heidelberg, Feb. 2000, pp. 245--258.

[27] J. Clark and P. C. Van Oorschot, "SoK: SSL and HTTPS: Revisiting Past Challenges and Evaluating Certificate Trust Model Enhancements," in *IEEE SnP*, Berkeley, USA, May 2013.

[28] M. Khodaei and P. Papadimitratos, "Scalable & Resilient Vehicle-Centric Certificate Revocation List Distribution in Vehicular Communication Systems," *IEEE Transactions on Mobile Computing (IEEE TMC)*, vol. 20, no. 7, pp. 2473--2489, Jul. 2021.

[29] ------, "The Key to Intelligent Transportation: Identity and Credential Management in Vehicular Communication Systems," *IEEE Vehicular Technology Magazine*, vol. 10, no. 4, pp. 63--69, Dec. 2015.

[30] V. Kumar, J. Petit, and W. Whyte, "Binary Hash Tree based Certificate Access Management for Connected Vehicles," in *Proceedings of the 10th ACM Conference on Security and Privacy in Wireless and Mobile Networks*, Boston, USA, Jul. 2017.

[31] 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, May 2018.

[32] M. Khodaei and P. Papadimitratos, "Efficient, Scalable, and Resilient Vehicle-Centric Certificate Revocation List Distribution in VANETs," in *ACM Conference on Security & Privacy in Wireless and Mobile Networks*, Stockholm, Sweden, Jun. 2018, pp. 172--183.

[33] M. A. Simplicio Jr, E. L. Cominetti, H. K. Patil, J. E. Ricardini, and M. V. M. Silva, "ACPC: Efficient Revocation of Pseudonym Certificates using Activation Codes," *Ad Hoc Networks*, Jul. 2018.

[34] J. G. Andrews, S. Buzzi, W. Choi, S. V. Hanly, A. Lozano, A. C. Soong, and J. C. Zhang, "What Will 5G Be?" *IEEE Journal on Selected Areas in Communications (JSAC)*, vol. 32, no. 6, pp. 1065--1082, 2014.

[35] M. Agiwal, A. Roy, and N. Saxena, "Next Generation 5G Wireless Networks: A Comprehensive Survey," *IEEE Communications Surveys & Tutorials*, vol. 18, no. 3, pp. 1617--1655, 2016.

[36] K. Abboud, H. A. Omar, and W. Zhuang, "Interworking of DSRC and Cellular Network Technologies for V2X Communications: A Survey," *IEEE Transactions on Vehicular Technology (TVT)*, vol. 65, no. 12, pp. 9457--9470, Jul. 2016.

[37] M. Khodaei, H. Jin, and P. Papadimitratos, "Towards Deploying a Scalable & Robust Vehicular Identity and Credential Management Infrastructure," in *IEEE Vehicular Networking Conference (VNC)*, Paderborn, Germany, Dec. 2014.

[38] H.-C. Hsiao, A. Studer, C. Chen, A. Perrig, F. Bai, B. Bellur, and A. Iyer, "Flooding-Resilient Broadcast Authentication for VANETs," in *ACM Mobile Computing and Networking*, Las Vegas, Nevada, USA, Sep. 2011.

[39] I. Ari, B. Hong, E. L. Miller, S. A. Brandt, and D. D. Long, "Managing Flash Crowds on the Internet," in *IEEE/ACM MASCOTS*, Orlando, FL, USA, Oct. 2003, pp. 246--249.

[40] Z. Ma, F. Kargl, and M. Weber, "Pseudonym-on-demand: A New Pseudonym Refill Strategy for Vehicular Communications," in *IEEE Vehicular Technology Conference*, Calgary, BC, Sep. 2008.

[41] M. Khodaei and P. Papadimitratos, "Evaluating On-demand Pseudonym Acquisition Policies in Vehicular Communication Systems," in *ACM MobiHoc Workshop on Internet of Vehicles and Vehicles of Internet (ACM IoV-VoI)*, Paderborn, Germany, Jul. 2016.

[42] J. R. Douceur, "The Sybil Attack," in *ACM Peer-to-peer Systems*, London, UK, Mar. 2002.

[43] B. Xiao, B. Yu, and C. Gao, "Detection and Localization of Sybil Nodes in VANETs," in *Proceedings of the 2006 workshop on Dependability issues in wireless ad hoc networks and sensor networks (DIWANS)*, Los Angeles, CA, USA, Sep. 2006, pp. 1--8.

[44] P. Golle, D. Greene, and J. Staddon, "Detecting and correcting malicious data in vanets," in *ACM VANET*, Philadelphia, PA, USA, Oct. 2004, pp. 29--37.

[45] M. Khodaei, H. Noroozi, and P. Papadimitratos, "Scaling Pseudonymous Authentication for Large Mobile Systems," in *Proceedings of the 12th ACM Conference on Security & Privacy in Wireless and Mobile Networks*, Miami, FL, USA, May 2019.

[46] H. Noroozi, M. Khodaei, and P. Papadimitratos, "DEMO: VPKIaaS: A Highly-Available and Dynamically-Scalable Vehicular Public-Key Infrastructure," in *Proceedings of the 2018 ACM conference on Security and privacy in wireless & mobile networks*, Stockholm, Sweden, Jun. 2018, pp. 302--304.

[47] P. Cincilla, O. Hicham, and B. Charles, "Vehicular PKI Scalability-Consistency Trade-Offs in Large Scale Distributed Scenarios," in *IEEE Vehicular Networking Conference (VNC)*, Columbus, Ohio, USA, Dec. 2016.

[48] P. Papadimitratos, V. Gligor, and J.-P. Hubaux, "Securing Vehicular Communications-Assumptions, Requirements, and Principles," in *Workshop on Embedded Security in Cars (ESCAR)*, Berlin, Germany, Nov. 2006.

[49] N. Bißmeyer, "Misbehavior Detection and Attacker Identification in Vehicular Ad-Hoc Networks," Ph.D. dissertation, Technische Universität, Dec. 2014.

[50] P. Papadimitratos, ""On the road" - Reflections on the Security of Vehicular Communication Systems," in *IEEE International Conference on Vehicular Electronics and Safety (ICVES)*, Columbus, OH, USA, Sep. 2008.

[51] J. Sermersheim, "Lightweight Directory Access Protocol (LDAP)," RFC 4511, Tech. Rep., Jun. 2006.

[52] E. Rescorlad, "The Transport Layer Security (TLS) Protocol Version 1.3," RFC 8446, Tech. Rep., Aug. 2018.

[53] "Google Cloud Platform," Jan. 2019. [Online]. Available: https://cloud.google.com/gcp/

[54] "Kubernetes: Production-Grade Container Orchestration," Jan. 2019. [Online]. Available: https://kubernetes.io/

[55] "Redis, In-memory Data Structure Store, Used as a Database," https://redis.io/, Oct. 2018.

[56] D. Cooper, S. Santesson, S. Farrell, S. Boeyen, R. Housley, and W. T. Polk, "Internet X. 509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile," RFC 5280, Tech. Rep., May 2008.

[57] "Cloud Storage Pricing," https://cloud.google.com/storage/pricing#price-tables, Aug. 2021.

[58] E. Rescorla and N. Modadugu, "Datagram Transport Layer Security V.1.2," Jan. 2012.

[59] C. Adams, S. Farrell, T. Kause, and T. Mononen, "Internet X.509 Public Key Infrastructure Certificate Management Protocol (CMP)," RFC 4210, Tech. Rep., Sep. 2005.

[60] B. F. Cooper, A. Silberstein, E. Tam, R. Ramakrishnan, and R. Sears, "Benchmarking Cloud Serving Systems with YCSB," in *ACM SoCC*, Indianapolis, Indiana, USA, Jun. 2010, pp. 143--154.

[61] "Google Cloud Armor," cloud.google.com/armor, Aug. 2021.

[62] "Best Practices for DDoS Protection and Mitigation on Google Cloud Platform," https://cloud.google.com/files/GCPDDoSprotection-04122016.pdf, Apr. 2016.

[63] "AWS Best Practices for DDoS Resiliency," https://d1.awsstatic.com/whitepapers/Security/DDoS_White_Paper.pdf, Jun. 2018.

[64] T. Aura, P. Nikander, and J. Leiwo, "DoS-Resistant Authentication with Client Puzzles," in *Proceedings of Security Protocols Workshop*, New York, USA, Apr. 2001.

[65] M. Abliz and T. Znati, "A Guided Tour Puzzle for Denial of Service Prevention," in *IEEE Annual Computer Security Applications Conference (ACSAC)*, Honolulu, HI, Dec. 2009, pp. 279--288.

[66] "SSL DDoS Attacks and How to Defend Against Them," www.link11.com/en/blog/threat-landscape/ssl-ddos-attacks-and-how-to-defend-against-them/, Aug. 2018.

[67] "TLS Security 6: Examples of TLS Vulnerabilities and Attacks," https://www.acunetix.com/blog/articles/tls-vulnerabilities-attacks-final-part/, Mar. 2019.

[68] "ProVerif: Cryptographic Protocol Verifier in the Formal Model," https://prosecco.gforge.inria.fr/personal/bblanche/proverif/, Feb. 2021, accessed Feb. 15, 2021.

[69] B. Blanchet, B. Smyth, V. Cheval, and M. Sylvestre, "ProVerif 2.02 pl1: Automatic Cryptographic Protocol Verifier, User Manual and Tutorial," https://prosecco.gforge.inria.fr/personal/bblanche/proverif/manual.pdf, Tech. Rep., Sep. 2020.

[70] B. Blanchet, "Automatic Proof of Strong Secrecy for Security Protocols," in *IEEE Symposium on Security and Privacy*, California, USA, May 2004.

[71] D. Dolev and A. Yao, "On the Security of Public Key Protocols," *IEEE Transactions on Information Theory*, vol. 29, no. 2, pp. 198--208, Mar. 1983.

[72] L. Hirschi, D. Baelde, and S. Delaune, "A Method for Verifying Privacy-type Properties: The Unbounded Case," in *IEEE Symposium on Security and Privacy (SP)*, San Jose, CA, USA, May 2016, pp. 564--581.

[73] A. Pfitzmann and M. Köhntopp, "Anonymity, Unobservability, and Pseudonymity - A Proposal for Terminology," in *Designing Privacy Enhancing Technologies*. Springer, Mar. 2001, pp. 1--9.

[74] D. Baelde, S. Delaune, and S. Moreau, "A Method for Proving Unlinkability of Stateful Protocols," in *IEEE Computer Security Foundations Symposium (CSF)*, Boston, US, Jan. 2020, pp. 169--183.

[75] "An Open Source Load Testing Tool," locust.io/, Jan. 2019.

[76] "Google Container Registry," https://cloud.google.com/container-registry/, Oct. 2018.

[77] "Google Kubernetes Engine v1.9.6," https://cloud.google.com/kubernetes-engine/, Oct. 2018.

[78] P. Heinlein, "FastCGI," *Linux journal*, vol. 1998, no. 55es, 1998.

[79] "XML-RPC for C/C++," http://xmlrpc-c.sourceforge.net/, 2018, accessed April 25, 2018.

[80] "Google Protocol Buffer," https://developers.google.com/protocol-buffers/, 2018, accessed April 25, 2018.

[81] P. Molloy, M. Khodaei, P. Hallgren, A. Thenorio, and P. Papadimitratos, "SecProtobuf: Implicit Message Integrity Provision in Heterogeneous Vehicular Systems," in *IEEE Vehicular Networking Conference (VNC)*, Ulm, Germany, November 2021.

[82] "YAML API Reference," https://learn.getgrav.org/advanced/yaml, Oct. 2018.

[83] "Amazon Web Services," https://aws.amazon.com/, Oct. 2018.

[84] "Cloud Memorystore," https://cloud.google.com/memorystore/, Jan. 2019.

[85] "Prometheus," https://prometheus.io/, Jan. 2019.

[86] "Grafana," https://grafana.com/, Jan. 2019.

[87] "Memorystore for Redis Pricing," https://cloud.google.com/memorystore/docs/redis/pricing, Sep. 2021.

[88] "Prometheus & Grafana: Google Cloud Marketplace," https://kubernetes.io/docs/tasks/run-application/horizontal-pod-autoscale/, Jan. 2019.

[89] S. Uppoor, O. Trullols-Cruces, M. Fiore, and J. M. Barcelo-Ordinas, "Generation and Analysis of a Large-scale Urban Vehicular

Mobility Dataset,'' *IEEE Transactions on Mobile Computing (TMC)*, vol. 13, no. 5, pp. 1061--1075, May 2014.

[90] L. Codeca, R. Frank, and T. Engel, ''Luxembourg SUMO Traffic (LuST) Scenario: 24 Hours of Mobility for Vehicular Networking Research,'' in *IEEE Vehicular Networking Conference (VNC)*, Kyoto, Japan, Dec. 2015.

[91] ''Horizontal Pod Autoscaler,'' https://kubernetes.io/docs/tasks/run-application/horizontal-pod-autoscale/, Jan. 2019.

[92] L. Fischer, A. Aijaz, C. Eckert, and D. Vogt, ''Secure Revocable Anonymous Authenticated Inter-vehicle Communication (SRAAC),'' in *Conference on Embedded Security in Cars (ESCAR)*, Berlin, Germany, Nov. 2006.

[93] F. Schaub, F. Kargl, Z. Ma, and M. Weber, ''V-tokens for Conditional Pseudonymity in VANETs,'' in *IEEE Wireless Communication and Networking Conference (WCNC)*, Sydney, Australia, Apr. 2010.

[94] M. Raya, P. Papadimitratos, I. Aad, D. Jungels, and J.-P. Hubaux, ''Eviction of Misbehaving and Faulty Nodes in Vehicular Networks,'' *IEEE Journal on Selected Areas in Communications (JSAC)*, vol. 25, no. 8, pp. 1557--1568, Oct. 2007.

[95] S. Ruj, M. A. Cavenaghi, Z. Huang, A. Nayak, and I. Stojmenovic, ''On Data-Centric Misbehavior Detection in VANETs,'' in *IEEE Vehicular Technology Conference (VTC)*, San Francisco, CA, USA, Sep. 2011, pp. 1--5.

[96] S. Reidt, M. Srivatsa, and S. Balfe, ''The Fable of the Bees: Incentivizing Robust Revocation Decision Making in Ad Hoc Networks,'' in *ACM CCS*, Chicago, Illinois, US, Nov. 2009.

[97] D. Goodin, ''New Hack on Comodo Reseller Exposes Private Data,'' May 2011.

[98] J. Leyden, ''Inside 'Operation Black Tulip': DigiNotar hack analysed,'' https://www.theregister.co.uk/2011/09/06/diginotar_audit_damning_fail/, Sep. 2011.

[99] N. McAllister, ''Browser makers rush to block fake Google.com security cert,'' https://www.theregister.co.uk/2013/01/04/turkish_fake_google_site_certificate/, Jan. 2013.

[100] J. Freudiger, M. Raya, M. Félegyházi, P. Papadimitratos, and J.-P. Hubaux, ''Mix-zones for Location Privacy in Vehicular Networks,'' in *Win-ITS*, Vancouver, BC, Canada, Aug. 2007.

[101] B. Wiedersheim, Z. Ma, F. Kargl, and P. Papadimitratos, ''Privacy in Inter-vehicular Networks: Why Simple Pseudonym Change is not Enough,'' in *Seventh international conference on wireless on-demand network systems and services (WONS)*, Kranjska Gora, Slovenia, Feb. 2010, pp. 176--183.

[102] M. Khodaei, H. Noroozi, and P. Papadimitratos, ''POSTER: Privacy Preservation through Uniformity,'' in *Proceedings of the 2018 ACM conference on Security and privacy in wireless & mobile networks*, Stockholm, Sweden, Jun. 2018, pp. 279--280.

[103] M. Khodaei and P. Papadimitratos, ''Poster: Mix-Zones Everywhere: A Dynamic Cooperative Location Privacy Protection Scheme,'' in *IEEE Vehicular Networking Conference (VNC)*, Taipei, Taiwan, Dec. 2018.

[104] C. Vaas, M. Khodaei, P. Papadimitratos, and I. Martinovic, ''Nowhere to hide? Mix-Zones for Private Pseudonym Change using Chaff Vehicles,'' in *IEEE Vehicular Networking Conference (VNC)*, Taipei, Taiwan, Dec. 2018.

[105] M. Khodaei and P. Papadimitratos, ''Cooperative Location Privacy in Vehicular Networks: Why Simple Mix-zones are not Enough,'' *IEEE Internet Of Things Journal*, vol. 8, no. 10, pp. 7985--8004, May 2021.

[106] S. Gisdakis, M. Laganà, T. Giannetsos, and P. Papadimitratos, ''SEROSA: SERvice Oriented Security Architecture for Vehicular Communications,'' Boston, MA, USA, Dec. 2013, pp. 111--118.

[107] ''cloud-based EJBCA,'' Feb. 2022. [Online]. Available: https://www.primekey.com/products/ejbca-cloud/

[108] ''Cloud PKI as a Service,'' Feb. 2022. [Online]. Available: https://www.keyfactor.com/platform/cloud-pki-as-a-service/

[109] ''DigiCert,'' Feb. 2022. [Online]. Available: https://www.digicert.com/

[110] ''Hydrantid,'' Feb. 2022. [Online]. Available: https://hydrantid.com/

[111] T. H.-J. Kim, L.-S. Huang, A. Perring, C. Jackson, and V. Gligor, ''Accountable Key Infrastructure (AKI): A Proposal for a Public-key Validation Infrastructure,'' in *ACM WWW*, Rio de Janeiro, Brazil, May 2013.

[112] L. Dykcik, L. Chuat, P. Szalachowski, and A. Perrig, ''BlockPKI: An Automated, Resilient, and Transparent Public-Key Infrastructure,'' in *IEEE International Conference on Data Mining Workshops (ICDMW)*, Singapore, Singapore, Nov. 2018, pp. 105--114.

[113] M. Khodaei, A. Messing, and P. Papadimitratos, ''RHyTHM: A Randomized Hybrid Scheme To Hide in the Mobile Crowd,'' in *IEEE Vehicular Networking Conference*, Torino, Italy, Nov. 2017.

Mohammad Khodaei earned his Ph.D degree from KTH Royal Institute of Technology, Stockholm, Sweden, in 2020. He is currently working at Scania Group as a Cyber Security Manager. His research interests include security and privacy in smart cities, the Internet of Things, in-vehicle communication systems, distributed systems, and cloud computing.

Hamid Noroozi earned his M.Sc in computer science at KTH Royal Institute of Technology, and currently working as a Senior DevOps engineer in the industry. As part of his M.Sc thesis, he has been collaborating with the Networked Systems Security group, under the supervision of Prof. Panos Papadimitratos.

Panagiotis (Panos) Papadimitratos earned his Ph.D. degree from Cornell University, Ithaca, NY, USA. At KTH, Stockholm, Sweden, he leads the Networked Systems Security group and he is a member of the steering committee of the Security Link center. He serves or served as: member (and currently chair) of the ACM WiSec conference steering committee; member of the PETS Editorial and Advisory Boards and the CANS conference steering committee; program chair for the ACM WiSec'16, TRUST'16 and CANS'18 conferences; general chair for ACM WISec'18, PETS'19 and IEEE EuroS&P'19 conferences; and Associate Editor of the IEEE TMC, ACM/IEEE ToN, IET IFS and ACM MC2R journals. Panos is a Fellow of the Young Academy of Europe, a Knut and Alice Wallenberg Academy Fellow, an IEEE Fellow, and an ACM Distinguished Member. His group web-page is: https://www.eecs.kth.se/nss.