Scaling Pseudonymous Authentication for Large Mobile Systems

ACM WiSec’19, May 17, 2019

Mohammad Khodaei, Hamid Noroozi, and Panos Papadimitratos

Networked Systems Security Group

www.eecs.kth.se/nss
Vehicular Communication Systems (VCS)

Illustration: C2C-CC
Secure Vehicular Communication Systems

VCS Security and Privacy

Basic Requirements
- Authentication & integrity
- Non-repudiation
- Authorization & access control
- Anonymity (conditional)
- Unlinkability (longer-term)
- Accountability

Vehicular Public-Key Infrastructure (VPKI)
- Ephemeral pseudonymous credentials
- Long-term credentials (Long Term Certificates (LTCs))
Vehicle-to-Vehicle (V2V)/Vehicle-to-Infrastructure (V2I) (V2X) message communications are digitally signed.

Messages are signed with the private key corresponding to the currently valid pseudonym.

Cryptographic operations in a Hardware Security Module (HSM)
Vehicular Public-Key Infrastructure (VPKI)
- Root CA (RCA)
- Long Term CA (LTCA)
- Pseudonym CA (PCA)
- Resolution Authority (RA)
- Lightweight Directory Access Protocol (LDAP)
- Roadside Unit (RSU)

- Vehicles registered with one LTCA (home domain)
- One or more PCA servers per domains
- Vehicles can obtain pseudonyms from any PCA (home or foreign domains)
- RCA or cross-certification
- Deanonymize (resolve pseudonyms) with the help of an RA
VPKI Challenges; Motivation

**Traditional PKI vs. Vehicular PKI**

- Dimensions (5 orders of magnitude more credentials)
- Complexity and constraints
  - Balancing act: security, privacy, and efficiency
    - *Honest-but-curious* VPKI entities
    - Performance constraints: safety- and time-critical operations (rates of 10 safety beacons per second)
  - Multiple and diverse entities, global deployment, long-lived entities
  - Cost-driven platform resource constraints
- Mechanics of revocation
  - Highly dynamic environment
  - Short-lived pseudonyms, multiple per entity
  - Need for efficient and timely distribution of Certificate Revocation Lists (CRLs)
  - Strong privacy protection prior to revocation events
Adversarial Model

- Honest-but-curious service providers

- Faulty PCAs could:
  - Issue multiple sets of (simultaneously valid) pseudonyms
  - Issue a set of pseudonyms for a non-existing vehicle
  - 'Incriminate' vehicles (users) during a pseudonym resolution process

- Faulty LTCAs could:
  - 'Incriminate' vehicles (users) during the resolution process
  - Issue fake authorization tickets for pseudonym acquisition process

- A faulty RA can continuously initiate a pseudonym validation process towards inferring user sensitive information
Multiple VPKI entities (servers) could collude

Malicious (compromised) VCS entities
  - Interval adversaries, i.e., On-Board Units (OBUs) could
    - Repeatedly request multiple simultaneously valid pseudonyms, attempting to become 'Sibyl nodes'
  - Mount a clogging Denial of Service (DoS) attack against the VPKI servers

External adversaries, i.e., unauthorized entities, could try to:
  - Mount a clogging DoS attack against the VPKI servers
System Model and Assumptions

Pseudonym acquisition overview in the home and foreign domains.

- **P1 & P2**: Requests could be user “fingerprints”: exact times of requests throughout the trip
- **P3**: Request intervals falling within “universally” fixed intervals $\Gamma_{P3}$; pseudonym lifetimes aligned with the PCA clock

Objectives

- Design, analyse, implement and evaluate the VPKI
  - Management of credentials: provisioning, revocation, resolution
  - Standard-compliant implementation

- Resilience to *honest-but-curious* and *malicious* VPKI entities

- Eradication of Sybil-based misbehavior (without degrading performance)

- Handling of unexpected demanding loads, while being cost-effective

- Scalability

- Efficient revocation and resolution
VPKI as a Service (VPKIaaS)

- Refactoring the source code of a state-of-the-art VPKI
- Fully automated procedures of deployment
- Migration to the cloud, e.g., Google Cloud Platform (GCP), Amazon Web Service (AWS), Microsoft Azure
- Health and load metrics used by an orchestration service to scale in/out accordingly
- Eradication of Sybil-based misbehavior when deploying multiple replicas without diminishing the efficiency of the pseudonym acquisition process
- Functionality enhancements
VPKI as a Service (VPKIaaS) Architecture

High-level Overview of VPKIaaS Architecture on the Cloud
VPKI as a Service (VPKIaaS) Architecture

High-level Overview of VPKIaaS Architecture on the Cloud
VPKI as a Service (VPKIaaS) Architecture

High-level Overview of VPKIaaS Architecture on the Cloud
Credential Acquisition in VPKIaaS System

Pseudonym Acquisition Process

1. \((H(Id_{pca} \parallel Rnd_{256}), t_s, t_e, LT_{C_v}, N, t)\)

2. \(IK_{tkt} \leftarrow H(LT_{C_v} || t_s || t_e || Rnd_{IK_{tkt}})\)

3. \(tkt \leftarrow (H(Id_{pca} || Rnd_{tkt}), IK_{tkt}, t_s, t_e)\)

4. \(Cert(LTC_{ltca}, tkt)\)

5. \((tkt_{\sigma_{ltca}}, N+1, t)\)

6. \((t_s, t_e, (tkt)_{\sigma_{ltca}}, \{(K_1^v)_{\sigma_{ltca}}, \ldots, (K_n^v)_{\sigma_{ltca}}\}, N', t_{\text{now}})\)

7. \(\text{Verify}(LTC_{ltca}, (tkt)_{\sigma_{ltca}})\)

8. \(Rnd_v \leftarrow \text{GenRnd}()\)

9. \(\text{Verify}(K_1^v, (K_1^v)_{\sigma_{ltca}})\)

10. \(RIK_{pv} \leftarrow H(IK_{tkt} || K_1^v || t_s^i || t_e^i || H^i(Rnd_v))\)

11. \(\zeta \leftarrow (SN_i, K_i^p, CRL_v, BF_{CRL_v}, RIK_{pv}, t_s^i, t_e^i)\)

12. \((P_i^v)_{\sigma_{pca}} \leftarrow \text{Sign}(Lkpca, \zeta)\)

13. \((\{(P_1^v)_{\sigma_{pca}}, \ldots, (P_n^v)_{\sigma_{pca}}\}, Rnd_v, N+1, t_{\text{now}})\)
LTCA Sybil Attack Mitigation

- Checking if a ticket was issued to the requester
- Updating the Redis database if not
- Invoking the ticket issuance procedure

PCA Sybil Attack Mitigation

- Checking if pseudonyms were issued to (the requester of) a given ticket
- Updating the Redis database if not
- Invoking the pseudonym issuance procedure
null
Pseudonym Request Validation (by the PCA using Redis)

1: procedure \texttt{VALIDATEPSEUDONYMREQ}(SN_{i_{kt}}) \\
2: \hspace{1em} (\text{value}^i) \leftarrow \text{RedisQuery}(SN_{i_{kt}}) \\
3: \hspace{1em} \textbf{if} \text{value}^i == \text{NULL OR value}^i == \text{False} \textbf{then} \\
4: \hspace{2em} \text{RedisUpdate}(SN_{i_{kt}}, \text{True}) \\
5: \hspace{2em} \text{Status} \leftarrow \text{IssuePsnyms}(\ldots) \\
6: \hspace{2em} \textbf{if} \text{Status} == \text{False} \textbf{then} \\
7: \hspace{3em} \text{RedisUpdate}(SN_{i_{kt}}, \text{False}) \\
8: \hspace{3em} \text{return} (\text{False}) \\
9: \hspace{2em} \text{else} \\
10: \hspace{3em} \text{return} (\text{True}) \\
11: \hspace{2em} \textbf{end if} \\
12: \hspace{1em} \text{else} \\
13: \hspace{2em} \text{return} (\text{False}) \\
14: \hspace{2em} \textbf{end if} \\
15: \textbf{end procedure} \\

\hspace{1em} ▶ Checking if pseudonyms were issued to the requester for a given ticket \\
\hspace{1em} ▶ If the key does not exist or the value is false (i.e., unused) \\
\hspace{1em} ▶ Updating the database, setting value to true (i.e., used) \\
\hspace{1em} ▶ Invoking pseudonym issuance procedure \\
\hspace{1em} ▶ Failure during the pseudonym issuance process \\
\hspace{2em} ▶ Reverting \text{SN}_{i_{kt}} to False \\
\hspace{2em} ▶ Pseudonym issuance failure \\
\hspace{3em} ▶ Pseudonym issuance success \\
\hspace{2em} ▶ Suspected Sybil attack
## Pseudonym Issuance Validation Process

### Pseudonym Issuance Validation Process (by the RA)

1. $V_j : P^i_v \leftarrow (SN^i_i, K^i_v, IK_{P^i_v}, t^i_s, t^i_e)$
2. $V_j : \zeta \leftarrow (P^i_v)$
3. $V_j : (\zeta)_{\sigma_v} \leftarrow \text{Sign}(P^i_v, \zeta)$
4. $V_j \rightarrow RA : (Id_{req}, (\zeta)_{\sigma_v}, t_{now})$
5. RA : Verify($P_v, (\zeta)_{\sigma_v}$)
6. RA : $\zeta \leftarrow (P^i_v)$
7. RA : $(\zeta)_{\sigma_{ra}} \leftarrow \text{Sign}(Lk_{ra}, \zeta)$
8. RA $\rightarrow$ PCA : $(Id_{req}, (\zeta)_{\sigma_{ra}}, LTC_{ra}, N, t_{now})$
9. PCA : Verify($LTC_{ra}, (\zeta)_{\sigma_{ra}}$)
10. PCA : $(tkt, Rnd_{IK_{P^i_v}}) \leftarrow \text{Resolve}(P^i_v)$
11. PCA : $\chi \leftarrow (SN_{P^i_v}, tkt_{\sigma_{ltca}}, Rnd_{IK_{P^i_v}})$
12. PCA : $(\chi)_{\sigma_{pca}} \leftarrow \text{Sign}(Lk_{pca}, \chi)$
13. PCA $\rightarrow$ RA : $(Id_{res}, (\chi)_{\sigma_{pca}}, N+1, t_{now})$
14. RA : Verify($LTC_{pca}, \chi$)
15. RA : $(SN_{P^i_v}, tkt_{\sigma_{ltca}}, Rnd_{IK_{P^i_v}}) \leftarrow \chi$
16. RA : Verify($LTC_{ltca}, tkt_{\sigma_{ltca}}$)
17. RA : $(H(Id_{PCA} || Rnd_{tkt}), IK_{tkt}, t^i_s, t^i_e, Exp_{tkt}) \leftarrow tkt$
18. RA : $H(IK_{tkt} || K^i_v || t^i_s || t^i_e || Rnd_{IK_{P^i_v}}) ?= IK_{P^i_v}$
Security and Privacy Analysis

✓ Communication integrity, confidentiality, and non-repudiation
  - Certificates, TLS and digital signatures

✓ Authentication, authorization and access control
  - LTCA is the policy decision and enforcement point
  - PCA grants the service
  - Security association discovery through LDAP

✓ Concealing PCAs, F-LTCA, actual pseudonym acquisition period
  - Sending $H(PCA_{id} \parallel Rnd_{256}, t_s, t_e, LTC_v)$ to the H-LTCA
  - PCA verifies if $[t'_s, t'_e] \subseteq [t_s, t_e]$

✓ Thwarting Sybil misbehavior
  - LTCA never issues valid tickets with overlapping lifetime (for a given domain)
  - Tickets are bound to specific PCAs
  - PCA keeps records of ticket usage
  - Suspicious requests instantaneously validated via the Redis Memorystore
  - Redis on a single thread; pipeline guaranteed to sequentially execute commands
Qualitative Analysis

Security and Privacy Analysis (cont’d)

- ✓ Single deviant PCA issuing multiple simultaneously valid pseudonyms, or issuing pseudonyms without any valid ticket
  - The RA efficiently validates pseudonyms without harming user privacy

- ✓ High availability and fault-tolerance
  - Benign failure: the Kubernetes master can kill the running (faulty) Pod and create a new Pod
  - High loads: the Kubernetes master scales out the Pods

- ✓ Distributed DoS (DDoS) attacks on the VPKIaaS system
  - Network-level protection; puzzles
Experimental Setup

**VPKI testbed**
- Implementation in C++, OpenSSL for cryptographic protocols & primitives, TLS and Elliptic Curve Digital Signature Algorithm (ECDSA)-256 (ETSI [TR-102-638] and IEEE 1609.2).
- FastCGI to interface Apache web-server; we use XML-RPC & Google Protocol Buffers

**VPKIaaS system**
- Built and pushed Docker images for LTCA, PCA, RA, MySQL, and Locust, an open source load testing tool, to the Google Container Registry
- Google Kubernetes Engine (GKE) v1.10.11
- Configured a cluster of five Virtual Machines (VMs) (n1-highcpu-32), each with 32 vCPUs and 28.8GB of memory

**VPKIaaS Memorystore**
- Redis; in-memory key-value data store
- MySQL

---

### Experiment Parameters

<table>
<thead>
<tr>
<th>Parameters</th>
<th>Config-1</th>
<th>Config-2</th>
</tr>
</thead>
<tbody>
<tr>
<td>Total number of vehicles</td>
<td>1000</td>
<td>100, 50,000</td>
</tr>
<tr>
<td>Hatch rate</td>
<td>1</td>
<td>1, 100</td>
</tr>
<tr>
<td>Interval between requests</td>
<td>1000-5000 ms</td>
<td>1000-5000 ms</td>
</tr>
<tr>
<td>Pseudonyms per request</td>
<td>100, 200, 300, 400, 500</td>
<td>100, 200, 500</td>
</tr>
<tr>
<td>LTCA memory request</td>
<td>128 MiB</td>
<td>128 MiB</td>
</tr>
<tr>
<td>LTCA memory limit</td>
<td>256 MiB</td>
<td>256 MiB</td>
</tr>
<tr>
<td>LTCA CPU request</td>
<td>500 m</td>
<td>500 m</td>
</tr>
<tr>
<td>LTCA CPU limit</td>
<td>1000 m</td>
<td>1000 m</td>
</tr>
<tr>
<td>LTCA HPA</td>
<td>1-40; CPU 60%</td>
<td>1-40; CPU 60%</td>
</tr>
<tr>
<td>PCA memory request</td>
<td>128 MiB</td>
<td>128 MiB</td>
</tr>
<tr>
<td>PCA memory limit</td>
<td>256 MiB</td>
<td>256 MiB</td>
</tr>
<tr>
<td>PCA CPU request</td>
<td>700 m</td>
<td>700 m</td>
</tr>
<tr>
<td>PCA CPU limit</td>
<td>1000 m</td>
<td>1000 m</td>
</tr>
<tr>
<td>PCA HPA</td>
<td>1-120; CPU 60%</td>
<td>1-120; CPU 60%</td>
</tr>
</tbody>
</table>

**Config-1:** normal vehicle arrival rate; every 1-5 sec, a new vehicle joins the system, requesting 100-500 pseudonyms.

**Config-2:** flash crowd scenario; on top of Config-1, 100 new vehicles join the system every 1-5 sec, requesting 100-200 pseudonyms.
Experimental Setup (cont’d)

Network connectivity
- Varies depending on the actual OBU-VPKI connectivity
- Reliable connectivity to the VPKI (e.g., RSU, Cellular, opportunistic WiFi)

Metrics
- End-to-end pseudonym acquisition latency from the initialization of ticket acquisition protocol till successful completion of pseudonym acquisition protocol
- High availability, robustness, reliability, dynamic-scalability

Use cases
- Large-scale pseudonym provision
- VPKIaaS with Flash Crowd Load Pattern
- Dynamic scalability of the VPKIaaS

Remark
- Pseudonyms issued with non-over-lapping intervals, to mitigate Sybil-based misbehavior
- Average daily commute 10-30 minutes (actual urban vehicular mobility dataset), or 1 hour (according to the US DoT)
- Obtaining 100 and 500 pseudonyms per day implies pseudonym lifetimes of 14.4 minutes or 3 minutes respectively, covering 24 hours trip duration
- Requesting pseudonyms based on Config-2, i.e., VPKIaaS system would serve 720,000 vehicles joining the system within an hour
**Performance Evaluation**

(a) CDF of end-to-end latency to issue a ticket

(b) CDF of end-to-end processing delay to issue pseudonyms

**Large-scale pseudonym acquisition (based on Config-1):**

- **End-to-end Latency for ticket:** $F_x(t = 24 \text{ ms}) = 0.999$.
- **Batch of 100 pseudonyms per request:** 99.9% of the vehicles are served within less than 77 ms ($F_x(t = 77 \text{ ms}) = 0.999$)
- **Batch of 500 pseudonyms per request:** $F_x(t = 388 \text{ ms}) = 0.999$
**Performance Evaluation (cont’d)**

![Graph showing CPU utilization and number of requests per second](image)

**Graph Description:**
- **CPU utilization and number of requests per second (100 pseudonyms per request):**

**VPKIaaS system in a flash crowd situation (based on Config-2):**

- CPU utilization hits a 60% threshold, services scale out, CPU utilization drops
- Latency to issue a single ticket is: \[ F_x(t = 87 \text{ ms}) = 0.999 \]
- Batch of 100 pseudonyms per request: \[ F_x(t = 192 \text{ ms}) = 0.999 \]
- ‘normal’ conditions vs. flash crowd: latency for a single ticket from 24 ms to 87 ms; latency for issuing 100 pseudonyms from 77 ms to 192 ms
Performance Evaluation (cont’d)

(e) Average e2e latency to obtain pseudonyms

(f) CDF of e2e latency, observed by clients (including the networking latency)

VPKIaaS system with flash crowd load pattern (based on Config-2):

- All vehicles obtained a batch of 100 pseudonyms within less than 4,900 ms

Note: The CR manuscript refers to improvement over prior implementation achieved by a standalone implementation [10]: latency for issuing a pseudonym \( \approx 4 \text{ms} \). The same figure for the VPKIaaS system is 0.56 ms (56 ms to issue 100 pseudonyms). The performance overall is captured by Figs. 4-7, which depict data for the VPKIaaS system.
Performance Evaluation (cont’d)

(g) Number of active vehicles and CPU utilization

(h) Dynamic scalability of VPKIaaS system

Reliability and dynamic scalability of the VPKIaaS system (based on Config-2):

- Each vehicle requests 500 pseudonyms (CPU utilization observed by HPA)
- Synthetic workload generated using 30 containers, each with 1 vCPU and 1GB of memory
Summary

- Refactored a state-of-the-art VPKI source code, with fully automated procedures of deployment and migration to the cloud.
- Health and load metrics used by an orchestration service to scale in/out accordingly.
- Eradicated Sybil-based misbehavior when deploying multiple replicas of a microservice, without diminishing the efficiency of the pseudonym acquisition.
- Enhanced features.
- Providing extensive experimental evaluation.
Summary (cont'd)

- Practical framework, issues on-demand pseudonyms for large-scale vehicular communication systems
- Highly efficient, scalable, and resilient
- Viable solution for deploying secure and privacy-protecting vehicular communication systems
- Investigating further adversarial behavior by the VPKI entities
- Investigating the performance of cryptographic operations on the Cloud-HSMs


Scaling Pseudonymous Authentication for Large Mobile Systems

ACM WiSec’19, May 17, 2019

Mohammad Khodaei, Hamid Noroozi, and Panos Papadimitratos

Networked Systems Security Group
www.eecs.kth.se/nss