A Cloud-native Vehicular Public Key Infrastructure

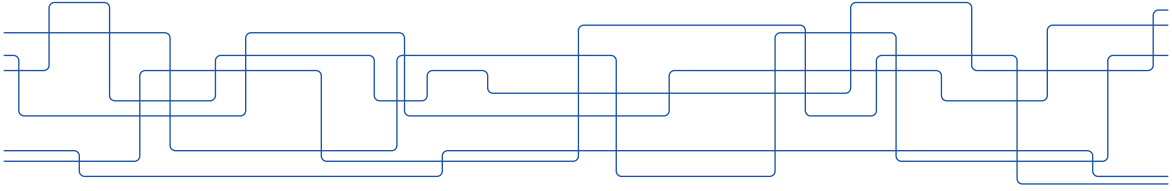
Towards a Highly-available and Dynamically-scalable VPKIaaS

Hamid Noroozi

Supervisor: Mohammad Khodaei

Examiner: Panos Papadimitratos

Networked Systems Security Group (NSS)
School of Electrical Engineering and Computer Science
KTH Royal Institute of Technology



Vehicular Communication Systems (VCSs)

Communication

- Vehicle-to-Vehicle (V2V)
- Vehicle-to-Infrastructure (V2I)

Messages

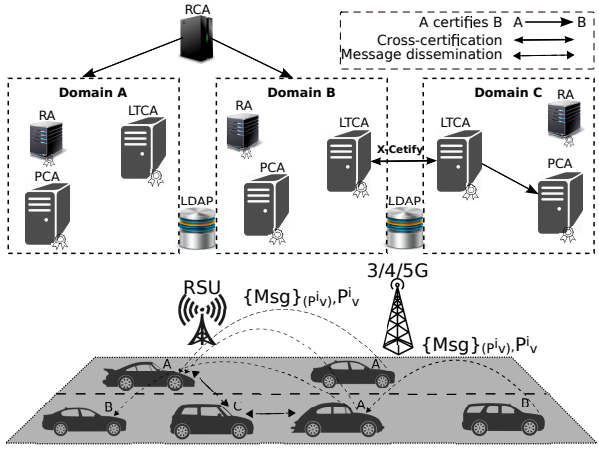
- Cooperative Awareness Messages (CAMs)
- Decentralized Environmental Notification Messages (DENMs)

Basic requirements

- Message authentication & integrity
- Message non-repudiation
- Authorization & access control
- Entity authentication
- Accountability
- Anonymity (conditional)
- Unlinkability (long-term)

Multi-domain Vehicular Public-Key Infrastructure (VPKI) overview

1



- Registration with Long Term CA (LTCA)
- Ticket acquisition from LTCA
- Pseudonym acquisition from Pseudonym CA (PCA)
- Inter-Domain trust by Root CA (RCA) or X-certification
- Revoke anonymity by of Resolution Authority (RA) & PCA & LTCA

[1] "SECMACE: Scalable and Robust Identity and Credential Management Infrastructure in Vehicular Communication Systems". In: IEEE TITS 19.5 (2018).

VPKI deployment challenges

VPKI vs. traditional PKI

- Dimension (5 orders of magnitude more credentials)
- Privacy (anonymity & unlinkability)
- Short-lived pseudonyms
- High availability
- Dynamic scalability

Preloading

- Overlapping psnyms $\rightarrow$ Sybil-based misbehavior
- Non-overlapping psnyms $\rightarrow$ Waste of psnyms
- Expensive revocation (not efficient)

On-demand

- Non-overlapping psnyms
- Efficient revocation
- Reliable connection
- Requires high availability
- Rush hour & flash crowd

Research question

How to achieve:

- High availability
- Dynamic scalability
- Fault tolerance & resilience
- Self-healing
- Large-scale deployment

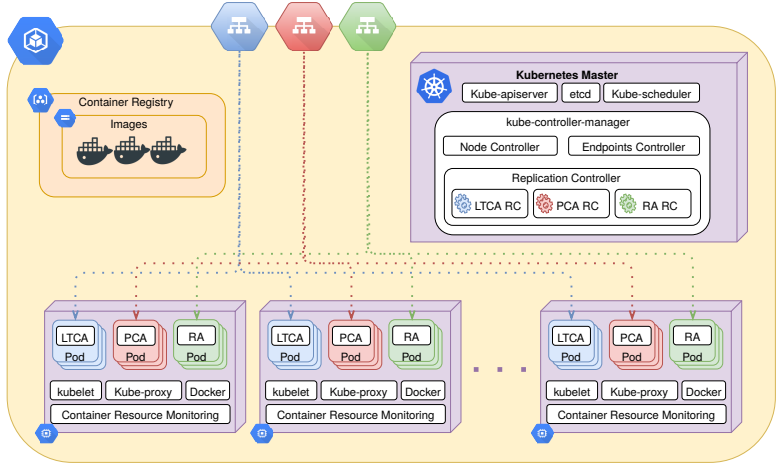
VPKI as a Service (VPKIaaS) Overview

Microservice architecture

- Refactoring VPKI
- Containerization
- Health metric
- Load metric

Kubernetes

- Google Kubernetes Engine (GKE)
- Automation
- Declarative language
  - Deployments
  - Services
  - Ingresses



Secret management

Key Management Service (KMS)

- Offered by Google Cloud Platform (GCP)
- Federal Information Processing Standard (FIPS) PUB 140-2 level 2 and/or 3
- Role Based Access Control (RBAC) provided by Identity & Access Management (IAM)
- Vendor lock-in
- Overhead for each cryptographic operation

Kubernetes secret management

- Secret volumes
- Cloud-agnostic
- More efficient than KMS
- No protection during deployment

Secret management (cont'd)

KMS + Kubernetes secret management

- Encrypt secret volumes
- Bootstrap with KMS
- RBAC provided by IAM
- No major overhead
- Minimize the impact of vendor lock-in

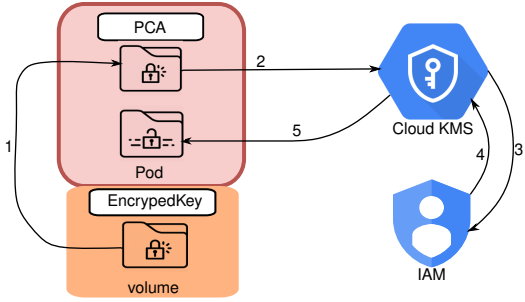


Figure: VPKIaaS bootstrapping secrets

1. Load encrypted key into the memory
2. PCA asks Cloud KMS to decrypt the key
3. Cloud KMS asks IAM for authorization control
4. IAM responds with yes/no based on RBAC
5. PCA receives the decrypted key, if authorized, otherwise it will be denied

Sybil attack while scaling horizontally

PCA/LTCA Operation

- Asynchronous
  - High performance
  - No Sybil attack protection
- Synchronous
  - Performance depends on the operation
  - Sybil attack protection possible

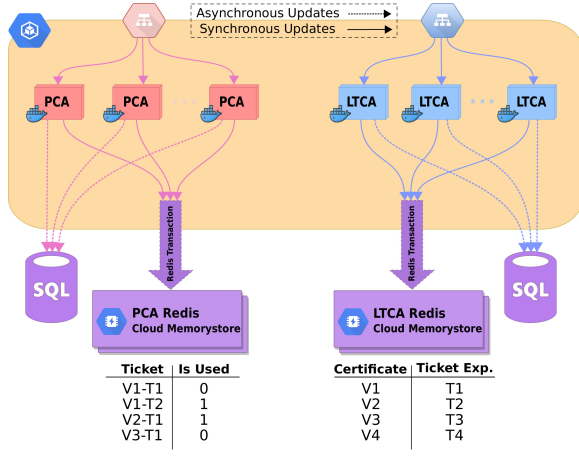


Figure: VPKIaaS Sybil attack prevention with Redis and MySQL

Sybil attack prevention by LTCA

Protocol 1 Ticket Request Validation

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* ← *IssueTicket*(. . .) ▷ Invoking ticket issuance procedure
6: **if** *Status* == *False* **then**
7: RedisUpdate(SN^i_{LTC}, $value^i$) ▷ Reverting 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**

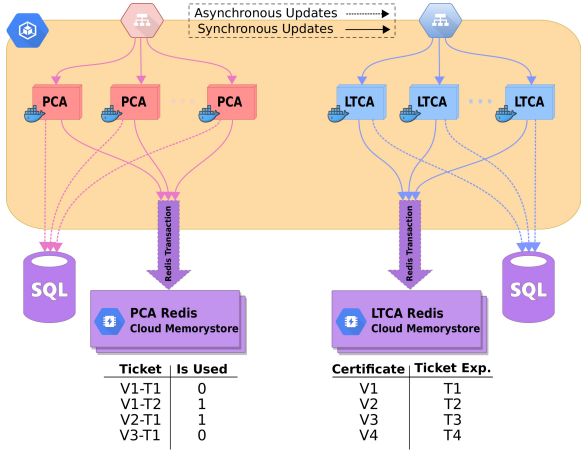


Figure: VPKIaaS Sybil attack prevention with Redis and MySQL

Sybil attack prevention by PCA

Protocol 2 Pseudonym Request Validation

1: **procedure** ValidatePseudonymReq(SN^i_{tkt})
2: $(value^i) \leftarrow$ 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**

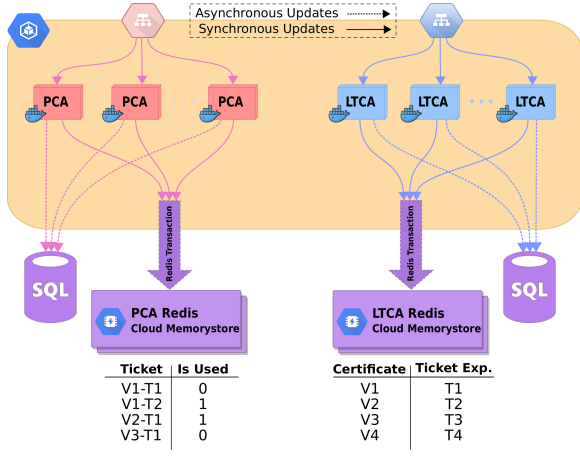


Figure: VPKIaaS Sybil attack prevention with Redis and MySQL

Prerequisite setup

Load generator

- Vehicle implementation
- Locust framework
- Locust interface for XML-RPC

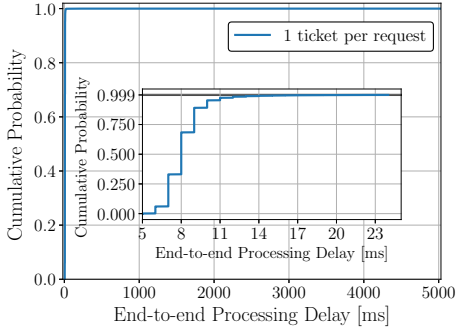
Monitoring

- Prometheus & Grafana
- Export data from Prometheus using Styx
- Monitoring Horizontal Pod Autoscaler (HPA)
- Monitoring Locust

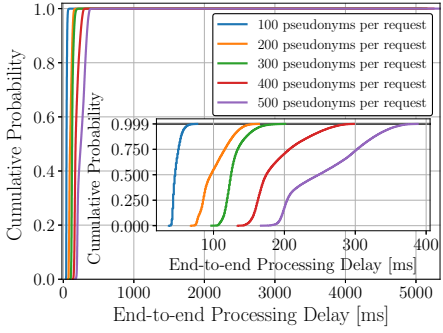
Parameters	Config-1	Config-2
total number of vehicles	1000	100, 50,000
hatch rate	1	1, 100
interval between requests	1000-5000 ms	1000-5000 ms
pseudonyms per request	100, 200, 300, 400, 500	100, 200, 500
LTCA memory request	128 MiB	128 MiB
LTCA memory limit	256 MiB	256 MiB
LTCA CPU request	500 m	500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

- Config-1: normal vehicle arrival rate; every second 1 vehicle simulator joins, every 1-5 sec all simulators simulate vehicles requesting 100-500 pseudonyms
- Config-2: flash crowd scenario; every second 100 vehicle simulators join, every 1-5 sec all simulators simulate vehicles requesting 100, 200, 500 pseudonyms

Performance Evaluation



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

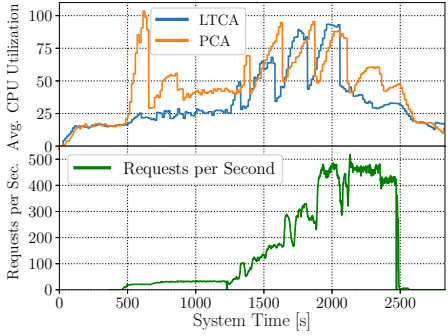


(b) CDF of end-to-end processing delay to issue pseudonyms

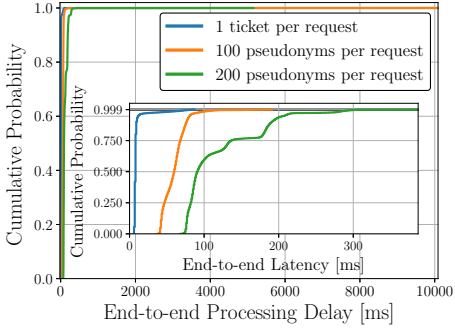
Large-scale pseudonym acquisition (based on Config-1)

- (a) End-to-end Latency for ticket: $F_x(t = 24\ ms) = 0.999$.
- (b) Asking for 100 pseudonyms per request, 99.9% of the vehicles are served within less than 77 ms ($F_x(t = 77\ ms) = 0.999$)
- (b) Asking for 500 pseudonyms per request,99.9% of the vehicles are served within less than 388 ms $F_x(t = 388\ ms) = 0.999$

Performance Evaluation (cont'd)



(c) CPU utilization and the number of requests per second (100 pseudonyms per request)

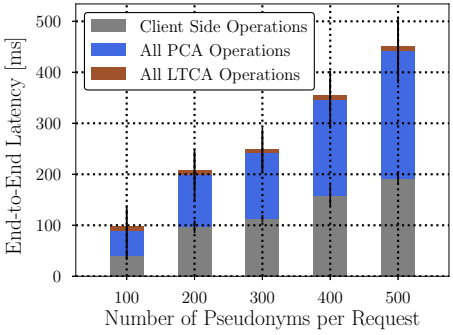


(d) CDF of processing latency to issue tickets and pseudonyms

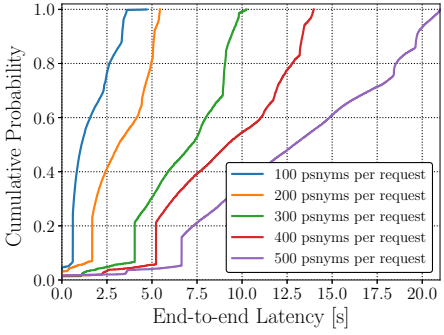
Flash crowd situation (based on Config-2)

- (c) CPU utilization hits 60% threshold, services scale out, CPU utilization drops
- (d) The processing latency to issue a single ticket is: $F_x(t = 87\ ms) = 0.999$
- (d) Issuing a batch of 100 pseudonyms per request: $F_x(t = 192\ ms) = 0.999$
- 'normal' conditions vs. flash crowd: processing latency of issuing a single ticket increases from 24 ms to 87 ms; the processing latency to issue a batch of 100 pseudonyms increased from 77 ms to 192 ms

Performance Evaluation (cont'd)



(e) Average e2e latency to obtain pseudonyms

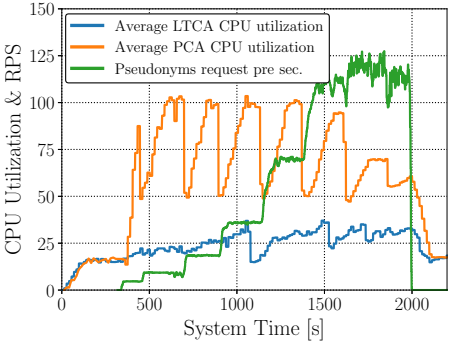


(f) CDF of e2e latency, observed by clients

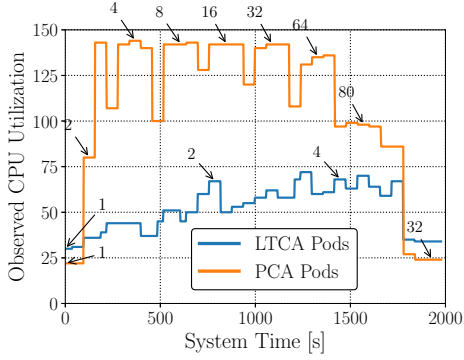
Flash crowd situation (based on Config-2)

- (e) The processing delay for issuing 100 psnyms is $\approx$ 56 ms which is 36-fold improvement comparing to 2010 ms reported in prior work [4]
- (f) During a surge of requests, *all* vehicles obtained a batch of 100 pseudonyms within less than 4,900 ms (including the networking latency)
- 100 vehicles join the system every second, but they simulate a new vehicle every 1-5 seconds. After all 50000 vehicles joined the system, every 1-5 seconds 50000 new vehicles join. After an hour at least $\approx$ 36 million vehicles will be served.

Performance Evaluation (cont'd)



(g) Number of active vehicles and CPU utilization



(h) Dynamic scalability of VPKIaaS system

Dynamic Scalability & High Availability (with flash crowd load pattern, based on Config-2)

- Each vehicle requests 500 pseudonyms
- Synthetic workload generated using 30 containers, each with 1 vCPU and 1GB of memory (based on Config-2)
- (h) CPU utilization observed by Horizontal Pod Autoscaler (HPA)
- Shows how our VPKIaaS system dynamically scales out or scales in according to the rate of pseudonyms requests.

Contribution summary

VPKI $\rightarrow$ VPKIaaS

- Refactoring state-of-the-art VPKI
- Microservices architecture
- Health & Load metrics
- Eradication of Sybil attacks against VPKIaaS
- Declarative deployment on Kubernetes
- Automated deployment on GCP

Performance evaluation

- Vehicle simulator
- XML-RPC for Locust
- Monitoring tools
- Various stress test scenarios (normal & flash crowd)

Future work

- Distributed DoS (DDoS) protection
  - Puzzle-based schemes similar to SECMACE.
  - Cloud Armor & Rule-based GCP Web Application Firewall (WAF)
- Secret management
  - Cloud Hardware Security Module (HSM)
  - Secrets OPerationS (SOPS)
- Service Mesh for microservices
  - mutual Transport Layer Security (TLS) (mTLS)
  - Geographically distributed multi-cluster
  - Federation of clusters
  - Domain Name System (DNS) weighted routing

Poster/Demo/Publications

- 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 (ACM WiSec), Miami, FL, USA, May 2019. [Online].
- H. Noroozi, M. Khodaei, and P. Papadimitratos. "VPKIaaS: Towards Scaling Pseudonymous Authentication for Large Mobile Systems," Cybersecurity and Privacy (CySeP) Summer School, June, 2019.
- H. Noroozi, M. Khodaei, and P. Papadimitratos, "DEMO: VPKIaaS: A Highly-Available and Dynamically-Scalable Vehicular Public-Key Infrastructure," in Proceedings of the ACM Conference on Security and Privacy in Wireless & Mobile Networks (WiSec), Stockholm, Sweden, June 2018.
- M. Khodaei, H. Noroozi, and P. Papadimitratos, "POSTER: Privacy Preservation through Uniformity," in Proceedings of the ACM Conference on Security and Privacy in Wireless & Mobile Networks (WiSec), Stockholm, Sweden, June 2018.
- H. Noroozi, M. Khodaei, and P. Papadimitratos. "VPKIaaS: A Highly-Available and Dynamically-Scalable Vehicular Public-Key Infrastructure," Cybersecurity and Privacy (CySeP) Summer School, June, 2018.
- M. Khodaei and P. Papadimitratos. "Security & Privacy for Vehicular Communication Systems," Cybersecurity and Privacy (CySeP) Summer School, June, 2018.
- H. Noroozi, M. Khodaei, and P. Papadimitratos. "A Highly Available and Dynamically Scalable Vehicular Public-Key Infrastructure (VPKI): VPKI as a Service (VPKIaaS)," Cybersecurity and Privacy (CySeP) Summer School, June, 2017.

CySeP 2019

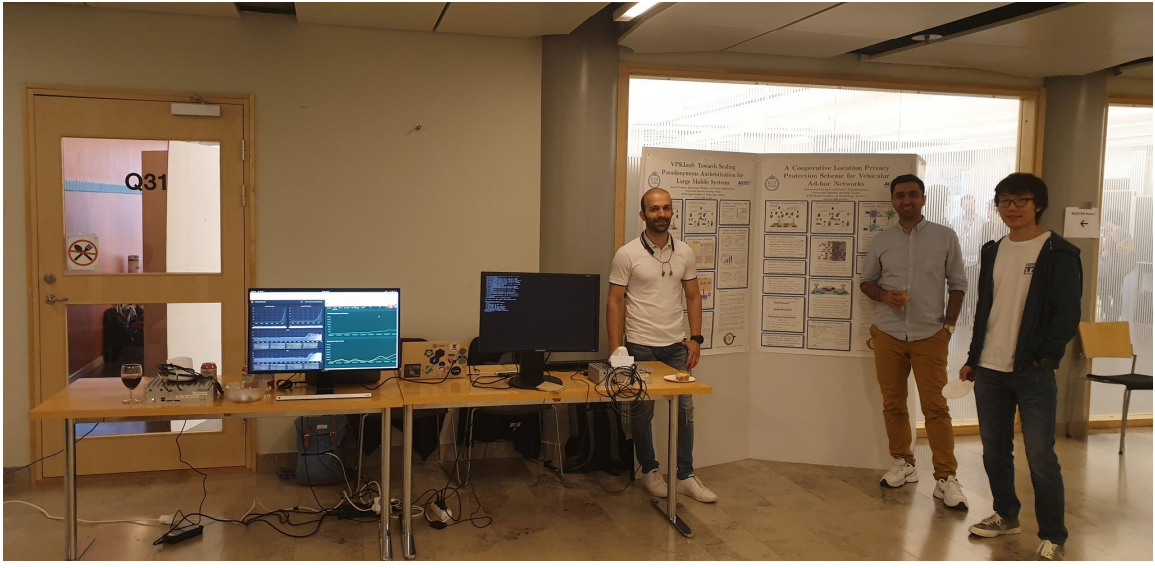


Figure: 180 million pseudonyms issued at CySeP summer school 2019

Bibliography I

[1] 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 (ACM WiSec)*, Miami, FL, USA, May 2019. [Online]. Available: https://dl.acm.org/doi/10.1145/3317549.3323410

[2] W. Whyte, A. Weimerskirch, V. Kumar, and T. Hehn, "A Security Credential Management System for V2V Communications," in *IEEE Vehicular Networking Conference (VNC)*, Boston, MA, Dec. 2013, pp. 1–8. [Online]. Available: https://ieeexplore.ieee.org/abstract/document/6737583

[3] M. Fowler and K. Beck, "Refactoring: Improving the design of existing code," 2018. [Online]. Available: https://martinfowler.com/books/refactoring.html

[4] 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. [Online]. Available: https://ieeexplore.ieee.org/abstract/document/7835970

[5] M. Abliz and T. Znati, "A Guided Tour Puzzle for Denial of Service Prevention," in *IEEE ACSAC*, Honolulu, HI, Dec. 2009, pp. 279–288. [Online]. Available: https://doi.org/10.1109/ACSAC.2009.33

[6] T. Aura, P. Nikander, and J. Leiwo, "DoS-Resistant Authentication with Client Puzzles," in *Proceedings of Security Protocols Workshop*, New York, USA, Apr. 2001. [Online]. Available: https://doi.org/10.1007/3-540-44810-1_22

[7] "Cloud Armor," Apr. 2021. [Online]. Available: https://cloud.google.com/armor

[8] "SOPS: Secrets OPerationS," Apr. 2021. [Online]. Available: https://github.com/mozilla/sops

[9] "What's a Linux container?" Mar. 2021. [Online]. Available: https://www.redhat.com/en/topics/containers/whats-a-linux-container

[10] "What is a Container?A standardized unit of software," Mar. 2021. [Online]. Available: https://www.docker.com/resources/what-container

[11] "Google Kubernetes Engine Overview," Feb. 2021. [Online]. Available: https://cloud.google.com/kubernetes-engine/docs/concepts/kubernetes-engine-overview

[12] "Kubernetes workloads, Pods," Feb. 2021. [Online]. Available: https://kubernetes.io/docs/concepts/workloads/pods/

[13] "Kubernetes workloads, Deployments," Feb. 2021. [Online]. Available: https://kubernetes.io/docs/concepts/workloads/controllers/deployment/

[14] "Kubernetes service," Feb. 2021. [Online]. Available: https://kubernetes.io/docs/concepts/services-networking/service/

[15] "Kubernetes ingress," Feb. 2021. [Online]. Available: https://kubernetes.io/docs/concepts/services-networking/ingress/

[16] "Kubelet," Feb. 2021. [Online]. Available: https://kubernetes.io/docs/reference/command-line-tools-reference/kubelet/

Bibliography II

[17] M. Fowler, "Microservices, a definition of this new architectural term," March 2014. [Online]. Available: https://martinfowler.com/articles/microservices.html

[18] J. R. Douceur, "The Sybil Attack," in *ACM Peer-to-peer Systems*, London, UK, Mar. 2002. [Online]. Available: https://link.springer.com/chapter/10.1007/3-540-45748-8_24

A Cloud-native Vehicular Public Key Infrastructure

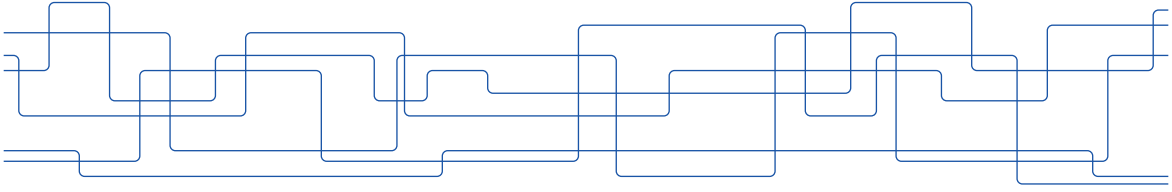
Towards a Highly-available and Dynamically-scalable VPKIaaS

Hamid Noroozi

Supervisor: Mohammad Khodaei

Examiner: Panos Papadimitratos

Networked Systems Security Group (NSS)
School of Electrical Engineering and Computer Science
KTH Royal Institute of Technology



Adversarial model

- Entities are *honest-but-curious*
- LTCA can:
  - Issue a fake/invalid ticket
  - Fraudulently accuse another vehicle
- PCA can:
  - Issue many psnyms, potentially all valid at the same time for a legit vehicle
  - Issue psnyms, for non-existing vehicle
  - Fraudulently accuse another vehicle
- VCS entities can:
  - Sybil attacks
  - DDoS attacks

VPKIaaS key concepts

- **Managed Service:** A service offered by a Managed Service Provider (MSP) via ongoing monitoring, maintenance and support for customers
- **Container:** A unit of packaged software along with its dependencies running as an isolated process
- **Docker:** A software facilitating build, shipment and running containers
- **Kubernetes:** An container orchestration platform
- **GKE:** A managed Kubernetes cluster offered by GCP
- **Pod:** The smallest unit of execution in Kubernetes which may contain one or more containers
- **Deployment:** A resource object in Kubernetes defining a Pod's life-cycle and its attributes

VPKIaaS key concepts (cont'd)

- **Service:** An abstract resource in Kubernetes defining a logical set of Pods, and the way they can be accessed
- **Ingress:** An Application Programming Interface (API) object at Kubernetes edge network handling external access to a service in cluster
- **Kubelet:** A primary agent of Kubernetes, running on worker nodes
- **Horizontal scalability:** The ability of increasing/decreasing capacity by adding/removing replicas, nodes to/from a system running the same software
- **Vertical scalability:** The ability of increasing/decreasing capacity by adding/removing hardware component to/from a system
- **Microservices architecture:** An architectural style for an application defining it as loosely coupled services that can scale in/out independently
- **Sybil attack:** Exploiting a system by creating more [pseudonymous] identities than one should and uses them to gain more influential advantage
- **Redis:** A high performance in-memory key-value data store.

Conclusion

- Practical framework for large-scale deployment of VPKIaaS
- High Availability with enterprise level Service Level Agreement (SLA)
- Resilient, fault tolerant and self-healing
- Resource efficiency through dynamic scalability
- Horizontal scalability without the risk of Sybil attacks