Secure Smartphone-based Registration and Key Deployment for Vehicle-to-Cloud Communications

Julian Timpner
timpner@ibr.cs.tu-bs.de

Dominik Schürmann
schuerm@ibr.cs.tu-bs.de

Lars Wolf
wolf@ibr.cs.tu-bs.de

Institute of Operating Systems and Computer Networks
TU Braunschweig
Braunschweig, Germany

ABSTRACT

Intelligent Transportation Systems that rely on Vehicle-to-Cloud communications are emerging. Of particular interest to us is the question how drivers can securely register their vehicle with cloud services and deploy keys for securing vehicular communications. We therefore present a secure smartphone-based registration and key deployment process for these applications. We combine an adapted OAuth flow for Smartphone-to-Cloud communications with a novel key deployment mechanism for Public Key Infrastructures in vehicular networks. The primary contributions of the proposed key deployment process are a high degree of independence from central authorities and feasible security audits due to an open protocol design. As an instance of future Intelligent Transportation System applications that facilitate Vehicle-to-Cloud communications, we utilize the V-Charge project as a running example to discuss potential attack scenarios and their mitigations.

Categories and Subject Descriptors

C.2.0 [**General**]: Security and protection; K.6.5 [**Security and Protection**]: Authentication

Keywords

Key Deployment; V2C Security; Trust; Bundle Protocol; ITS; Smartphone

1. INTRODUCTION

Today, automobiles are becoming increasingly "smart", that is, equipped with advanced sensor systems and connected with their environment and thus able to participate in Intelligent Transportation System (ITS) applications. The European V-Charge Project[1] seeks to make a contribution to this challenging field by developing autonomous driving capabilities for electric cars including a parking and charging management system that provides automated valet parking and coordinated recharging strategies for integration into a future transportation system.

A V-Charge service, called ParkingManager (PM) [1], is deployed to each parking lot or garage that is using the V-Charge system. The PM is responsible for managing parking resources by allowing users to make parking reservations and to provide their vehicles with mission control information, starting with their arrival at the parking lot. Since each parking garage has its own PM, there is a central back-end service that connects all PMs (via Internet) and allows users to register with the V-Charge system which takes care of authentication. In order to cope with the challenging network conditions of the vehicular domain (e.g., intermittent connectivity) and the V-Charge scenario [1] (e.g., no vehicular Internet access), we facilitate a Disruption- or Delay-Tolerant Networking (DTN) architecture [2] to establish robust and secure Vehicle-to-Cloud (V2C) communications between the vehicles and the PM. For this purpose, we utilize IBR-DTN[2] [3], a lightweight and modular Bundle Protocol [4] implementation. Although we use DTN in our running example, the principles of our work can be applied to IP-based communications with only minor modifications.

Because of the current rise of smartphones and their rapid integration into our daily life, the V-Charge Android app is the primary tool for users to interact with the system. Its features include reservation making, status checking, and issuing drop-off and pick-up commands which trigger the local PM to take control of the vehicle or to send it to the pick-up zone, respectively. It does not serve as a remote control or access token and is not part of the DTN.

In this paper, we focus on the security challenges of such an ITS. Of particular interest to us is the question how drivers can securely register their vehicle with the V-Charge service and deploy keys for securing V2C communications. By means of a smartphone-based registration and key deployment process for V2C communications, we are able to achieve a high degree of user independence from third parties, since nobody but the owner (not even the OEM) ever possesses the vehicle's private key. Further, our open and easily auditable protocol warrants user trust in the underlying cryptographic principles. Although we propose a solution aiming at the V-Charge project, our concepts are more generally applicable. For instance, any vehicular cloud service relying on a Public Key Infrastructure (PKI) could use the same process, since we provide an application-indepen-

[1] `http://www.v-charge.eu`

[2] `http://www.ibr.cs.tu-bs.de/projects/ibr-dtn/`

dent means for secure key deployment without entrusting the service provider or a third party with the private key. Moreover, the proposed solution is applicable to vehicles that come pre-deployed with the required communication technologies as well as refitted ones.

The remainder of this paper is structured as follows. We discuss the current state-of-the-art of the field in Section 2. We present our security architecture in Section 3. The issues of secure Smartphone-to-Cloud communications are addressed in Section 4. We then propose the registration process and smartphone-based key deployment in Section 5. A discussion of the proposed system under security considerations is given in Section 6. Finally, we conclude in Section 7.

2. RELATED WORK

To the best of our knowledge, most vehicular key management research that facilitates some kind of PKI assumes that the keys have somehow been securely deployed to the vehicle, for instance by the OEM [5, 6] or a vehicle service station [7]. Others do not consider this point at all [8]. In contrast to the C2C-CC[3] PKI, for instance, our approach focuses on minimizing the trust required in central authorities for key generation/deployment on a per-service base. It consists of a process to generate/deploy keys to a vehicle's Hardware Security Module (HSM) using a smartphone, under full user control.

Our idea, however, shares some commonalities with car immobilizer systems. While most car immobilizer systems are based on proprietary protocols [9], Busold et al. [10] present an open security framework. Vehicle owners are equipped with NFC-enabled smartphones to authenticate against compatible HSMs employed in cars. Replacing dedicated transponders, smartphones can be used for a more flexible authentication system that allows enhanced features, such as delegation of access to guest drivers for a specific time. Besides the default host on the mobile device, a Trusted Execution Environment (TrEE) is required. Access tokens to unlock the car are generated by the manufacturer with an Authenticate-then-Encrypt scheme using an authentication secret and encryption key pre-deployed on the car.

Similar to Busold et al. [10], we utilize an NFC-enabled HSM to store the keys securely. In contrast to the above-mentioned immobilizer system, though, we utilize the smartphone as a trusted interim device to perform registration and key generation/deployment, instead of serving as a key replacement. As mentioned above, we neither rely on pre-deployed keys, nor a specialized TrEE. Since exact hardware requirements are beyond the scope of this paper, we refer to existing comparisons of available HSMs for vehicles [11].

3. SECURITY ARCHITECTURE

Besides local PMs communicating securely with vehicles via DTN, the V-Charge service consists of several central components: (a) the CustomerManager, (b) the Authorization Server, and (c) the DTN Certificate Authority (CA).

(a) The CustomerManager [1] provides registration functionality and all necessary service methods (pick-up, drop-off, status checking) via RESTful Web services that are being interfaced by the V-Charge smartphone application. All methods of the interface are protected by the OAuth 2.0 standard [12]. (b) A central OAuth Authorization Server handles all OAuth sessions and provides verification methods. The connection between smartphone and the RESTful Web services itself is secured by SSL. (c) The DTN CA manages certificates deployed to all participants in the DTN, i.e., PMs and vehicles.

[3] `http://www.car-2-car.org`

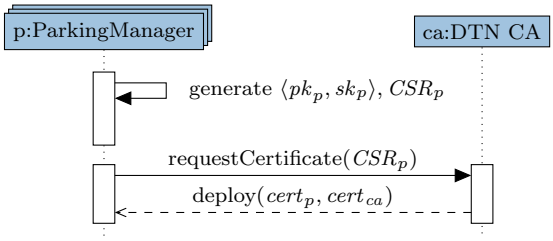

Figure 1: Deployment of certificates issued for local Parking Managers

As depicted exemplarily in Figure 1, a public/private key pair $\langle pk_p, sk_p \rangle$ is generated by each PM p. The DTN CA processes p's Certificate Signing Request (CSR) after it physically received it from p and sends back an certificate $cert_p$ signed with the CA's secret key sk_{ca}. After $cert_p$ has been deployed on p, vehicles equipped with the DTN CA's root certificate $cert_{ca}$ can verify if Bundles are affiliated to a PM (i.e., signed with sk_p) and are thus allowed to issue control commands (i.e., drop-off, pick-up). Contrary to vehicle certificates, PM certificates installed during their physical deployment contain a "ParkingManager" flag issued by our CA to allow differentiation between inter-vehicle Bundles and Bundles coming from servers acting as PMs. The connection between PM servers and central V-Charge services, such as DTN CA, is secured by a VPN.

3.1 Security in DTN

The experimental RFC 6257 [13] describes ciphersuites, Security Blocks, security policies, and how to deal with fragmented Bundles. Bundle Authentication Blocks (BABs) are defined to provide hop-by-hop integrity, which can be verified by every relaying node along the (multi-hop) path to the destination. End-to-end security is achieved by using the Payload Integrity Block (PIB) for authenticity and the Payload Confidentiality Block (PCB) for confidentiality.

In our scenario every participating node needs to be able to verify if an incoming Bundle is affiliated with V-Charge. Our goal is to restrict DTN communication to nodes participating in V-Charge. Thus, Bundles are always encrypted first using the PCB, and then signed using PIB to achieve end-to-end security. We configure IBR-DTN to accept/send PIB-signed and PCB-encrypted Bundles only. In order to thwart black/gray hole attacks, we facilitate hop-by-hop verification. To this end, BABs only support authentication, while PIBs allow signatures for verification using public keys. However, instead of using BABs or PIBs to verify relayed Bundles, we utilize IBR-DTN's support for SSL on the TCP convergence layer. A major advantage of SSL over BABs is that, if a node gets compromised, only its certificate needs to be revoked, instead of having to replace the symmetric BAB key across all nodes. Moreover, when Bundles are relayed between nodes, SSL allows to receive authenticated acknowledgments when peers accept Bundles sent by neighboring nodes in contrast to the sole use of PIBs.

3.2 Certificate Revocation

Certificate Revocation in DTNs is one of the interesting problems of DTN security. Naïve implementations require an additional check whether the certificate is currently valid when relaying/receiving a bundle. This can be done by querying a central database or by distributing Revocation Lists, whereas both solutions are problematic in networks with high delays and disruptions.

However, in the V-Charge scenario, vehicles periodically are in physical proximity to PMs or other vehicles, from which they are able to receive Revocation Lists via DTN. We propose to provide Revocation Lists as Bundles sent by the local PMs and signed by the central DTN CA to allow verification using $cert_{ca}$. These Revocation List Bundles are sent and updated with period δ and have a lifetime of $\delta + x$, where x defines a transition time that allows new revocation lists to invalidate old ones. Because all vehicles participating in V-Charge are relaying and storing currently valid Bundle Revocation Lists while parking, these lists can be kept as "Floating Content" [14]. Thus, arriving vehicles immediately receive currently valid lists either from the PM or from other parked vehicles. By verifying the signature, the lists' validity can be checked.

4. SMARTPHONE-TO-CLOUD SECURITY

As well as in DTN, security techniques for Smartphone-to-Cloud communication should be based on established and well-studied standards. Yet, client applications, such as the V-Charge Android app, impose different requirements regarding authentication compared to V2C communications. Most importantly, usability is a key requirement in mobile applications and heavily depends on the underlying security implementation. Like in most modern mobile applications, the hassle of requiring the user to repeatedly login before usage should be prevented, resulting in unlimited user sessions on mobile devices. Thus, similar to certificate revocation in V2C communications, revocation of user sessions is a crucial requirement. Considering a stolen smartphone, the user should be able to revoke access from this device in a convenient manner, before a thief can unwarrantably use V-Charge. Conclusively, the actual V-Charge password should never be stored on the mobile device, while providing revocable user sessions per device.

4.1 OAuth

We chose to incorporate the OAuth 2.0 standard [12]. Although its name is derived from Authentication [15], it provides Authorization for client applications. In our specific scenario it authorizes our smartphone application to access the central V-Charge service by issuing an *access_token*. This *access_token* is sent instead of an email/password combination on every request. Hence, no password needs to be stored on mobile devices. Due to V-Charge's centralized infrastructure and our specific attack scenarios, which are further described in Section 6.1, we implemented a well-defined subset of the OAuth 2.0 standard. Along with our use case, this adapted OAuth flow is applicable in a wide range of centralized scenarios, as described in the following.

A typical OAuth flow is based on redirecting users between an OAuth Provider and other HTTP services [12]. Authorization codes are issued by the resource owner, before redirecting the user back to the client. These authorization codes can then be used as a grant type to get authorization with access tokens via the Authorization Server. This grant type is designed to shift the user's trust regarding the login away from third-party services to the OAuth Provider.

In contrast to a typical social HTTP service (e.g., Twitter), V-Charge is a self-contained system with no public API for third-party clients. This means that all client applications are developed by the project itself, maintaining control and trust regarding the implementation of the actual login process (Authentication). Thus, instead of using an authorization code based grant type, we allow direct authentication with user credentials. Conclusively, V-Charge's OAuth implementation is based on a Token Endpoint only and allows the grant type *password* to retrieve a *refresh_token* and an *access_token* on first login. The grant type *refresh_token* serves to query a new *access_token*, which is valid for 1 hour, using the long-living *refresh_token*.

Revocation of a device's access to the V-Charge service is possible by logging into V-Charge's Web interface and deleting the refresh token of the lost/stolen devices. Keep in mind that revocation is only possible using the secret password of a user, which is not stored on the device itself. Further, changing a user password is only possible by entering the old password, not by using a *refresh_token*, making it impossible for a thief of a device to overtake a V-Charge account as a whole.

4.2 SSL Considerations

The connection between smartphone and the V-Charge Web service interface is secured using SSL. Due to recent security breaches in CAs [16], we implemented certificate checking without using a list of trusted CAs, provided by the mobile operating system. Instead, we rely on Certificate Pinning, meaning that all interfaces are secured using a certificate signed by a single private key, which is kept offline at a secure location. It allows us to shift the trust from outsourced CA management to our smartphone application by including the public key in the binary. Conclusively, all secure connections are verified with this key only.

In contrast to the interface used by the smartphone application, the V-Charge Web site will be secured by a certificate issued by a traditional CA to allow verification by general purpose browsers, until better standards [17, 18] are adopted.

5. REGISTRATION AND KEY DEPLOYMENT

Our proposal has two main objectives. First, private keys should always be generated and managed by the user itself, so there is no need to trust a service to handle key generation correctly, e.g., without storing or even worse leaking them. Second, all cryptographic operations should be auditable by sophisticated users, i.e., they are part of an open protocol.

Our process to register a new vehicle for usage with the V-Charge system is designed to be executed on off-the-shelf mobile devices which already provide a platform sufficiently trusted by its owner. Because the vehicle's DTN stack is configured to only send/receive signed and encrypted Bundles, it denies all incoming connections in its delivery condition, as neither keys nor certificates are pre-deployed. After successfully finishing the registration using our smartphone application by entering user credentials and additional infor-

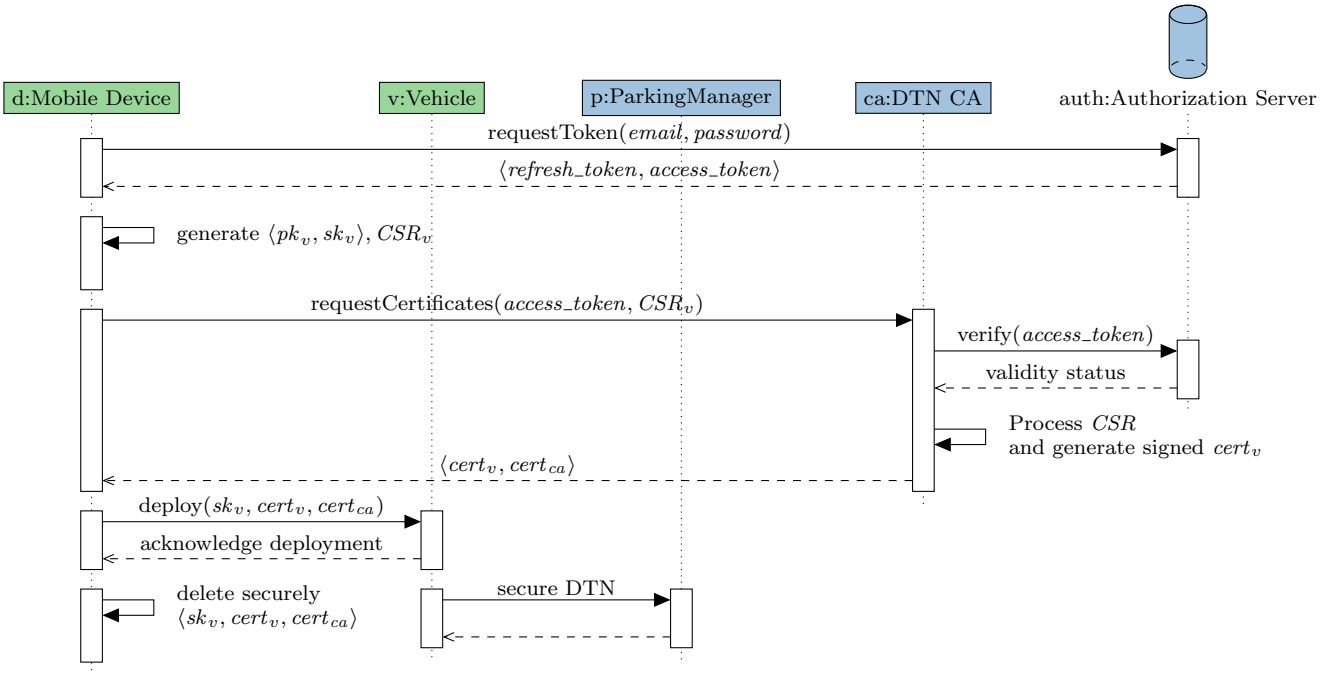

Figure 2: Key Deployment in V-Charge with the aid of smartphones to allow key generation on mobile devices, which are owned and inherently trusted by customers themselves.

mation, potentially a manual identification step, e.g., using Postident[4], finalizes the registration process.

5.1 Key Deployment

Following Figure 2, the user is now able to log into the V-Charge system using the chosen $\langle email, password \rangle$ combination. This is done by requesting a long-living *refresh_token* from our OAuth Authorization Server and sending the credentials over an SSL connection (cf. Section 4). Besides this, the device d also obtains an *access_token*, valid for 1 hour. As described in RFC 6749 [12], the *refresh_token* can be used to retrieve a new *access_token*.

Vehicles are uniquely identified by their Vehicle Identification Number *VIN*. The *VIN* of newly assembled vehicles, which come equipped with the required communication technology, can be directly added to the V-Charge database alongside the owner on purchase. To register refitted or resold vehicles for an account, customers are required to drive to a V-Charge accredited service station. Employees manually check the VIN_v and store it as a new database entry belonging to the customer. Alternatively, other proofs of possessing a specific vehicle with corresponding *VIN*, like digital possession certificates, could be accepted to overcome the manual verification step. The (optional) manual verification of VIN_v is the only step executed by employees of the service. The remaining key generation and deployment is solely done by the owner of the vehicle.

She starts the key generation process by using the Vehicle Registration Wizard, implemented in the smartphone application. After entering the vehicle to register, identified by the VIN_v, a key pair $\langle pk_v, sk_v \rangle$ is generated offline on the mobile device d. A corresponding CSR_v, including pk_v and VIN_v, is built, signed by sk_v and sent to V-Charge's DTN CA. While the connection is secured by SSL, the authorization and user affiliation is checked by validating the enclosed *access_token* against the Authorization Server.

The DTN CA queries the vehicle database $\mathcal{V}$ of an authorized user, containing VINs and issued certificates:

$$\mathcal{V} = \{\langle VIN_1, cert_1 \rangle, \ldots, \langle VIN_n, cert_n \rangle\}.$$

CSR_v is processed if $\langle VIN_v, \epsilon \rangle \in \mathcal{V}$, i.e., the VIN exists in V-Charge's database and no currently valid certificate is available. Our DTN CA generates $cert_v$ by signing the public key pk_v contained in CSR_v. Besides $cert_v$, the root certificate $cert_{ca}$ of our CA is returned.

Finally, $\langle sk_v, cert_v, cert_{ca} \rangle$, which are stored on the mobile device, must be deployed on the vehicle. This is done by utilizing NFC: An HSM is installed inside the vehicle along with the DTN stack that allows to deploy new root certificates or $\langle sk_v, cert_v \rangle$-pairs. The HSM is designed to only accept certificates that are issued for the *VIN* of the corresponding vehicle. Only providing a deploy and reset functionality, it does not allow to retrieve private keys and is sufficiently secure against attacks trying to extract keys. A PIN, which is stored offline by the vehicle owner, is needed to access the API of the HSM. After successful deployment, $\langle sk_v, cert_v, cert_{ca} \rangle$ is deleted from the mobile device in a secure way. For a short discussion of hardware requirements and assumptions see Section 6.2.

The DTN stack in the vehicle is now able to communicate securely with PM servers and to verify if incoming Bundles belong to the V-Charge DTN by checking if certificates corresponding to attached signatures are signed by $cert_{ca}$. Every outgoing Bundle is signed using sk_v, so that other nodes will accept the Bundle because of its signature which is only valid for V-Charge participants.

[4] `http://www.deutschepost.de/dpag?xmlFile=1015469`

6. DISCUSSION

Our proposed process provides a means to physically deploy cryptographic keys to a vehicle for use in a vehicular PKI. In the used example, we facilitate DTN as the underlying technology. Yet, the same principles can be applied to IP-based communications, for example by replacing the Bundle Security Protocol with IPSec. We achieve a high degree of independence from central authorities and third parties, since only the vehicle owner ever possesses its private key. Hence, we provide a solution to a trust issue in vehicular PKIs that was previously left open.

Our system can be put into practice without dependencies on specialized mobile devices and with few requirements regarding HSMs for vehicles (cf. Section 6.2). This makes it usable for real world use cases. If DTN hardware is available, our registration and key deployment is solely based on in-app wizards to register accounts and vehicles to lower the acceptance threshold for new users. Car manufacturers can decide to sell vehicles with pre-configured DTN stacks (without pre-deployed keys or certificates), while it is also feasible to upgrade existing vehicles with HSMs and communications hardware.

While we use the V-Charge system as a running example, other services can easily adopt our key deployment process to install their certificates. To extend the system for multi-service use, we suggest to provide a user interface to switch between different service providers and their certificates to restrict the DTN stack to one provider at a time and prevent conflicting communications.

6.1 Attack scenarios

We discuss possible attack scenarios from the point of view of an attacker T, while t denotes the vehicle of attacker T.

Steal sk_v before deployment: An attacker deploys a Trojan on the user's device by tricking her into installing software or by exploiting a vulnerability. To extract sk_v from the storage of the V-Charge application, the attacker probably needs to exploit the operating system for privilege escalation.

We argue that it is difficult to execute such attacks as sk_v is stored only for a short duration on the mobile device, e.g., for a couple of hours, until the user deploys the key to the vehicle. Thus, an attacker needs to know in advance that a user will register for the V-Charge system. On Android, these type of attacks are directed towards the well-studied Unix filesystem permissions, as each application storage is separated using unique Unix users [19, 20]. To strengthen smartphone security, a TrEE can be utilized, like it is required for solutions designed as key replacements [10].

Intercept $\langle email, password \rangle$: To intercept passwords, a keylogger needs to be installed before using the V-Charge app for the first time, as passwords are never stored, because our implementation is based on OAuth refresh tokens. This attack is almost as difficult to execute as the previous one, because it needs to exploit vulnerabilities to gain system permissions and interfere with other applications by logging their keystrokes.

Another possibility is to intercept the connection between smartphone and server. By using Certificate Pinning for the SSL connection, the possibility to execute Man-in-the-Middle (MitM) attacks with forged certificates using an overtaken CA can be excluded. Thus, the SSL standard itself needs to be attacked, which we consider beyond the scope of typical attackers.

Intercept *access_token*: As described before, this would require attacking the SSL standard itself.

Extract *refresh_token*: A Trojan is required to extract tokens from Android's *AccountManager*[5]. Because the *AccountManager* is specifically designed to store passwords and access tokens, this attack also requires significant privilege escalation exploits.

Another attack vector is the theft of the mobile device to gain physical access. This also includes unnoticed physical access, while the user is distracted. While this allows an attacker to read the whole flash storage, the *refresh_token* can be extracted. As this is actually the most realistic and common attack scenario to access user data, we implemented a revocation mechanism by logging into the V-Charge service and revoking the device's access.

Reset credentials in the HSM: As our HSM allows resetting and deployment of key pairs/ certificates, an attacker could prevent the vehicle from operating with the V-Charge system by resetting the storage of the HSM. This attack requires physical access to the HSM and previous theft of the PIN needed to access it. Thus, we argue that this attack is difficult to execute, while easier forms of attacks on availability exists, e.g., stabbing the tires.

Eavesdropping on NFC: We assume a Diffie-Hellman key exchange, followed by an AES encrypted transmission. Hence, by using the NFC-SEC standard [21] or an according implementation on the application layer, sk_v cannot be intercepted.

Relay attacks on NFC: Since the transmission of sk_v from the smartphone to the vehicle only happens once, as opposed to vehicular access control systems, the risk of a relay attack is low. Additionally, a manual confirmation step on the smartphone makes this attack even harder.

Register vehicle with made up *VIN*: As we require a manual verification step of the *VIN* by driving to a V-Charge service station or by providing alternative forms of owner verification, the attacker needs to fake the *VIN* or collude with V-Charge employees. Registering vehicles using a *VIN* previously not verified, is not possible as the *CSR* will be rejected by our CA.

Deploy attacker's $\langle sk_t, cert_t \rangle$ to a victim's vehicle: When the attacker has legitimately registered a vehicle, the retrieved pair $\langle sk_t, cert_t \rangle$ could be deployed to a victim's vehicle using physical access to the HSM. This attack is prevented as our HSM only accepts $cert_t$ if it is issued for the corresponding VIN_v of the vehicle.

Extract $\langle sk_v, cert_v \rangle$ from a victim's vehicle: This requires hacking the HSM, which we assume to be not feasible by average attackers due to discussed assumptions on hardware. However, if sophisticated attacks are observed that successfully extract these, a revocation of $cert_v$ has to be initiated and the owner of v is forced to re-generate sk_v and request a new $cert_v$.

6.2 In-Vehicle Key Storage

To securely store private keys on the vehicle, an HSM is required. While specific hardware implementation details are beyond the scope of this paper, some design specifications are assumed. In the automotive domain, HSMs should

[5] `http://developer.android.com/reference/android/accounts/AccountManager.html`

be especially durable and long-living. Our DTN stack delegates the decrypt/sign operations to the HSM, while encrypt/verify can still be executed by the stack itself. This way, sk_v is never exposed. To protect sk_v even further, a viable option for future implementations might be to let the keys directly be generated by the HSM. In comparison to smartphone-based key generation though, this would require an additional step to extract the generated pk_v prior to requesting $cert_v$, as well as a more complex user interaction. The API should only consist of a mode to reset its memory and a deployment mode to store new $\langle sk_v, cert_v \rangle$-pairs besides root certificates. On vehicle sale, the new owner is able to put it back into delivery condition using the reset mode. As vehicles are equipped with HSMs by service stations or car manufacturers, they need to be configured to only accept $\langle sk_v, cert_v \rangle$-pairs issued for the vehicle's *VIN*. We assume that a PIN is required to access the API, while NFC is proposed to serve as the communication protocol. As shown by previous research, MitM attacks on NFC are unfeasible [22].

7. CONCLUSIONS

We presented a novel approach for securely deploying cryptographic keys to vehicles in order to establish secure Vehicle-to-Cloud communications. By combining an adapted OAuth flow with a smartphone-based key deployment mechanism we achieve a high degree of independence from central authorities and third parties. In particular, only the vehicle owner ever possesses its private key. Hence, we provide a solution to a trust issue in vehicular PKIs that was previously left open. Moreover, our registration and key deployment process makes conducting security audits feasible due to an open protocol design. With the exception of storing the deployed keys in an actual HSM, the proposed system has been implemented and is currently under evaluation. We outlined how to deploy our system for general V2C services besides the used running example. Further, we discussed attack scenarios and how they can be mitigated.

8. ACKNOWLEDGMENTS

The project has received funding from the European Community's Seventh Framework Programme (FP7/2007-2013) under Grant Agreement Number 269916.

References

[1] J. Timpner and L. Wolf. "A Back-end System for an Autonomous Parking and Charging System for Electric Vehicles". In: *IEEE International Electric Vehicle Conference*. Greenville, SC, Mar. 2012, pp. 693–700.

[2] *Delay-Tolerant Networking Architecture*. RFC 4838 (Informational). Internet Engineering Task Force, Apr. 2007.

[3] S. Schildt, J. Morgenroth, W.-B. Pöttner, and L. Wolf. "IBR-DTN: A lightweight, modular and highly portable Bundle Protocol implementation". In: *Electronic Communications of the EASST* 37 (Jan. 2011).

[4] *Bundle Protocol Specification*. RFC 5050 (Experimental). Internet Engineering Task Force, Nov. 2007.

[5] 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". In: *Communications Magazine, IEEE* 46.11 (2008), pp. 100–109.

[6] S. Busanelli, G. Ferrari, and L. Veltri. "Short-lived key management for secure communications in VANETs". In: *11th International Conference on ITS Telecommunications*. ITST. 2011, pp. 613–618.

[7] A. Studer, E. Shi, F. Bai, and A. Perrig. "TACKing Together Efficient Authentication, Revocation, and Privacy in VANETs". In: *6th Annual IEEE Communications Society Conf. on Sensor, Mesh and Ad Hoc Communications and Networks*. SECON '09. 2009.

[8] A. Wasef, R. Lu, X. Lin, and X. Shen. "Complementing Public Key Infrastructure to Secure Vehicular Ad Hoc Networks [Security and Privacy in Emerging Wireless Networks]". In: *Wireless Communications, IEEE* 17.5 (2010), pp. 22–28.

[9] S. Tillich and M. Wójcik. "Security Analysis of an Open Car Immobilizer Protocol Stack". In: *Trusted Systems*. Ed. by C. Mitchell and A. Tomlinson. Vol. 7711. Lecture Notes in Computer Science. Springer Berlin Heidelberg, 2012, pp. 83–94.

[10] C. Busold, A. Taha, C. Wachsmann, A. Dmitrienko, H. Seudié, M. Sobhani, and A.-R. Sadeghi. "Smart Keys for Cyber-Cars: Secure Smartphone-based NFC-enabled Car Immobilizer". In: *Proc. of the Third ACM Conference on Data and Application Security and Privacy*. CODASPY '13. ACM, 2013, pp. 233–242.

[11] M. Wolf and T. Gendrullis. "Design, Implementation, and Evaluation of a Vehicular Hardware Security Module". In: *Information Security and Cryptology - ICISC 2011*. Ed. by H. Kim. Vol. 7259. Lecture Notes in Computer Science. Springer Berlin Heidelberg, 2012, pp. 302–318.

[12] D. Hardt. *The OAuth 2.0 Authorization Framework*. RFC 6749 (Proposed Standard). Internet Engineering Task Force, Oct. 2012.

[13] S. Symington, S. Farrell, H. Weiss, and P. Lovell. *Bundle Security Protocol Specification*. RFC 6257 (Experimental). Internet Engineering Task Force, May 2011.

[14] J. Ott, E. Hyytiä, P. Lassila, T. Vaegs, and J. Kangasharju. "Floating Content: Information Sharing in Urban Areas". In: *IEEE International Conference on Pervasive Computing and Communications*. PerCom. Seattle, USA, Mar. 2011.

[15] E. Hammer-Lahav. *The OAuth 1.0 Guide - History*. [Online; accessed 30-June-2013]. July 2011. URL: `http://hueniverse.com/oauth/guide/history`.

[16] *CAcert Wiki - CA Risk History*. [Online; accessed 30-June-2013]. 2013. URL: `http://wiki.cacert.org/Risk/History`.

[17] D. Wendlandt, D. Andersen, and A. Perrig. "Perspectives: Improving SSH-style Host Authentication with Multi-Path Probing". In: *Proc. USENIX Annual Technical Conference*. Boston, MA, June 2008.

[18] M. Marlinspike. *Trust Assertions for Certificate Keys*. Ed. by T. Perrin. Internet-Draft. Internet Engineering Task Force, Jan. 2013.

[19] Android Open Source Project. *Android Security Overview*. [Online; accessed 30-June-2013]. 2013. URL: `https://source.android.com/tech/security`.

[20] T. Vidas, D. Votipka, and N. Christin. "All Your Droid Are Belong to Us: A Survey of Current Android Attacks". In: *Proc. of the 5th USENIX Conference on Offensive Technologies*. USENIX Association. 2011.

[21] ISO. *Information technology – Telecommunications and information exchange between systems – NFC Security – Part 2: NFC-SEC cryptography standard using ECDH and AES*. ISO/IEC 13157-2 and ECMA-386. 2010.

[22] E. Haselsteiner and K. Breitfuß. "Security in Near Field Communication (NFC)". In: *Workshop on RFID Security*. RFIDSec. 2006.