Efficient and Secure Storage of Private Keys for Pseudonymous Vehicular Communication

Michael Feiri
Distributed and Embedded Security Research Group
University of Twente
The Netherlands
m.feiri@utwente.nl

Jonathan Petit
Distributed and Embedded Security Research Group
University of Twente
The Netherlands
j.petit@utwente.nl

Frank Kargl
Institute of Distributed Systems
University of Ulm
Ulm, Germany
frank.kargl@uni-ulm.de

ABSTRACT

Current standardization efforts for cooperative Intelligent Transportation Systems both in the U.S. and Europe foresee vehicles to use a large number of changeable pseudonyms for privacy protection. Provisioning and storage of these pseudonyms require efficient and secure mechanisms to prevent malicious use of pseudonyms. In this paper we investigate several techniques to improve secure and efficient storage of pseudonyms. Specifically, we propose schemes based on Physical Unclonable Functions (PUFs) that allow to replace expensive secure key storage by regular unsecured memory and still provide fully secure pseudonyms storage.

Categories and Subject Descriptors

C.2.0 [**Computer Communication Networks**]: General—*Security and protection*; K.6.5 [**Security and Protection**]: Physical security

General Terms

Security, Privacy

Keywords

PUF, HSM, Secure Storage, KDF, VANET, ITS

1. INTRODUCTION

In a near future, Vehicle-to-Vehicle (V2V) and Vehicle-to-Infrastructure (V2X) communication will enable vehicles to exchange information regarding safety, traffic condition and infotainment with each other. The On-Board Unit (OBU) will play a key role in these systems, as it manages incoming/outgoing messages and also performs security and privacy functions. Security and privacy of V2X communications are mandatory to enable a successful deployment of Intelligent Transportation System (ITS). As many V2X applications have a potential impact on traffic safety, their communication must be secured for obvious reasons [19]. Only messages from authenticated vehicles should be processed by receiving vehicles to prevent, for example, false safety-related warnings. Current standardization efforts, both in the U.S. and Europe, foresee that ITS will require the establishment of a Public Key Infrastructure (PKI), which manages trust and certificates in the ITS. The current set of standards [10, 11, 12] mandates the use of Elliptic Curve Digital Signature Algorithm (ECDSA) with P-256 elliptic curve for message authentication. A naïve implementation of authentication mechanisms breaks user privacy as every receiver learns the identity of the sender. Therefore, a pseudonymous credential – short *pseudonym*– should be implemented in order to prevent authentication to facilitate direct vehicle identification. One single pseudonym is not enough to ensure a sufficient level of privacy. Instead, this pseudonym has to be changed frequently, and even then, a powerful attacker may be able to track vehicles [41]. A central question here is how many such pseudonyms a vehicle possesses and how often it would have to contact the PKI for renewal. In general, a frequent connection to the PKI to renew pseudonyms cannot be guaranteed because large-scale coverage by road-side units (RSU) or cellular communication in every vehicle is considered unrealistic during early years of V2X deployment. Thus, the OBU has to store a potentially large set of pseudonyms to allow frequent change of pseudonyms in absence of backend connectivity. In a worst case, a vehicle would only be able to load new pseudonyms during (bi-)annual inspections in a garage. Recent research estimates that an OBU is required to store 105,120 pseudonyms for one year with each pseudonym valid for five minutes [15].

Each of those pseudonyms consists of a public-private key pair and a corresponding certificate and especially the private key needs to be stored securely to not compromise security of the overall system. If an attacker acquires access to the secret keys stored in a vehicle, she could perform sybil attacks, spoofing attacks, and in general jeopardize the authentication and privacy of the victim. In consequence, it must be guaranteed that a private key is strictly secured during all events in its life cycle. This goal can be achieved by designing systems to securely create, manage and destroy (private) keys, maintaining an audit trail of every operation executed during their existence. Hardware Security Modules (HSMs) [7] are specifically designed to protect private keys. HSMs are specialized tamper-proof devices in which cryptographic functions and embedded software properly manage keys and control their life cycles. They are designed in such a way that if an unauthorised attempt to access them

is made, this is considered an attempt to tamper and all critical internal parameters and keys are destroyed. The HSM features make them a crucial component in automotive platform security [22, 27]. However, HSMs are especially expensive if implemented on an FPGA [42], and a secure storage within an HSM adds complexity to the overall system. With ECDSA P-256 curve, the private keys of the one-year pseudonyms set proposed in [15] would require 256 bits $\times$ 105,120 = 3.2 Mbytes of secure storage – not considering yet any overhead for data management. This requirement is too high as current solutions offer a maximum of 512 kbytes [31, 1]. Therefore, we aim at trading secure storage of cryptographic key material for regular storage (i.e. outside of the HSM).

Contribution. In our previous work [35], we already investigated how Physical Unclonable Functions (PUFs) can be used for efficient and secure generation of private keys in the context of V2X pseudonym generation. In this paper, we focus on the aspect of efficient and secure key storage. We also provide a detailed overhead analysis of the five proposed schemes and compare their resilience against attacks.

Organization. The paper is organized as follows: Section 2 introduces the related work that deals with provisioning and secure storage of pseudonyms in vehicular network. Section 3 gives the necessary background on PUF, and describes our system and attacker model. Then, we investigate the classic solutions for secure storage in Section 4. Section 5 presents our approach how to use Physical Unclonable Function (PUF) for secure storage of V2X pseudonyms. We then compare classic secure storage and PUF-based secure storage solutions in Section 6. Finally, Section 7 draws some conclusions and outlines future work.

2. RELATED WORK

A number of related publications have proposed ways to enhance the provisioning to and storage of pseudonyms in vehicles. We first describe the method of pseudonym provisioning based on PKI that underlies current standardization efforts. Secondly, we detail autonomous pseudonym provisioning techniques that aim at reducing the key material to be stored. Thirdly, we give an overview on how large amount of keys are stored in modern operating systems.

2.1 PKI-based pseudonym provisioning

The conservative PKI-based approach is limited to optimize the provisioning process of pseudonyms. This can be achieved by optimizing the trade-off between availability requirements and storage requirements. The allocation can happen on-demand, if a connection from the vehicle to a Certificate Authority (CA) is available, or by caching preallocated pseudonyms in local secure storage. In this context a pseudonym is defined as the pair of a public and private key as well as the related certificate issued by a trusted CA in the PKI.

To the best of our knowledge, detailed analysis of this tradeoff do not exist. But a common assumption is to locally store a one year supply of pseudonyms inside vehicles [15, 3] to allow reloading new pseudonyms to vehicles during annual inspections if no other means of communication with the PKI backend exist. For resupplying pseudonyms it is expected that vehicles will be able to intermittently use IP connectivity through road side units, residential WiFi or mobile phone networks.

2.2 Autonomous pseudonym provisioning

Some approaches allow vehicles to generate new pseudonyms themselves without interaction with a backend.

One example are systems using a group signature scheme [3] to allow groups of vehicles to autonomously create key pairs for self-provisioned pseudonyms. These certificates validate the membership of a vehicle in a group, but do not reveal which member owns a particular pseudonym. Enrollment and removal of group members alter the group key material. This likely requires frequent rekeying and constant availability of the group manager, which can limit the appeal of this scheme. On the other hand, this scheme requires significantly less private key material to be stored. One valid set of long-term credentials for enrolling into an existing group or forming a new group is sufficient to bootstrap the system and create pseudonyms. Pseudonyms including private keys can be generated on demand and do not need to be stored beforehand.

Attribute-based authentication [29, 5, 43] allows a vehicle to generate pseudonyms entirely by itself, backed by a preshared cryptographic authorization attribute. A zero knowledge proof is performed between sender and receiver to verify the authenticity of a pseudonym without revealing any further identifying information about the signer. Similar to group signatures, only a set of long-term credentials for the zero knowledge proof is required to be stored. A vehicle can independently create pseudonyms, including private keys, on-demand.

The aforementioned pseudonym schemes have a distinctive set of advantages and disadvantages. Group-based signatures and attribute-based authentication systems do not need to store large amounts of private keys for pseudonyms. Unfortunately, the utility of these systems is limited in practice by interactive protocols, reachability requirements for authorities, or slow bilinear pairing based cryptographic primitives. Moreover, as vehicles autonomously generate pseudonyms, the number of pseudonyms available to one vehicle per time cannot be limited. Hence, large-scale Sybil attacks become possible [44]. These limitations make the schemes unsuitable for practical applications contexts with low latency requirements, such as vehicular communication.

2.3 Scalable secure key storage

Solutions for the secure local storage of large amounts of key material do exist in the form of password managers [40, 38], and more generically, in encrypted files [23] or filesystems [14]. These solutions store sets of keys and passwords in encrypted data stores, which are protected by a master secret, and additionally, by common operating system security features, such as access control lists.

Such solutions are effective with a human user in the loop to provide the master secret. In vehicular security it is not

expected that a human user will provide a password or similar authentication data that could be used to unlock an encrypted data store. Instead, an On-Board Unit needs direct access to all the required data to boot up into a fully operational state. Effectively the decryption keys have to be stored together with the encrypted data. This implies that this type of solution cannot work as an self-sufficient secure storage system. Though, encryption of private keys can be part of a solution that employs another type of secure storage for the master secret. Such a solution is described in Section 4.2.

3. SYSTEM MODEL

In this section we describe the system model considered in this paper. First, we explain Physical Unclonable Functions (PUFs) and related concepts. Then, we show how our solutions fit into the general OBU architecture. Finally, we describe the attacker model considered and discuss in Section 6 the level of protection offered by our solutions against such attackers.

3.1 Physical Unclonable Functions

A Physical Unclonable Function (PUF), as introduced in [32, 33], is a primitive that is bound to a physical system and extracts a pseudorandom bit string for key generation by mapping a set of challenges C_i to a set of responses R_i. This challenge-response behavior is highly dependent on the physical properties of the device in which the PUF is contained or embedded. PUFs consist of two parts:

i) a physical part, which is an intractably complex physical system that is very difficult to clone. It inherits its unclonability from uncontrollable process variations during manufacturing. For PUFs on an Integrated Circuit (IC), these process variations are typically deep-sub-micron variations such as doping variations in transistors.

ii) an operational part, which corresponds to the function.

In order to turn the physical system into a *function*, a set of challenges C_i (stimuli) has to be available to which the system responds with a set of sufficiently different responses R_i. The function can only be evaluated using the physical system and is unique for each physical instance because of process variations. Moreover, it is unpredictable even for an attacker with physical access.

PUF responses are noisy by nature. This means, that two calls to a single PUF with the same challenge c_i will output two different but closely related responses r_i, r_i'. The measure of closeness can be defined via a distance function, e.g., the Hamming distance. This distance function should be small for responses from the same device and very large for PUF responses from different devices. Since the plain PUF responses are noisy, they cannot be used as a key. In order to derive reliable and uniform data from (imperfect) sources of randomness, such as a PUF, the concept of a fuzzy extractor [8] or helper data algorithm [28] was introduced. Thus, we obtain a *master secret* from the fuzzy extractor. This master secret can be the seed for a key generation scheme to derive public/private key pair(s) which can then be used as a pseudonym(s). Alternatively the master secret can be used to first seed a key derivation scheme, which results in a larger amount of data that can then be used as seeds for key generation processes.

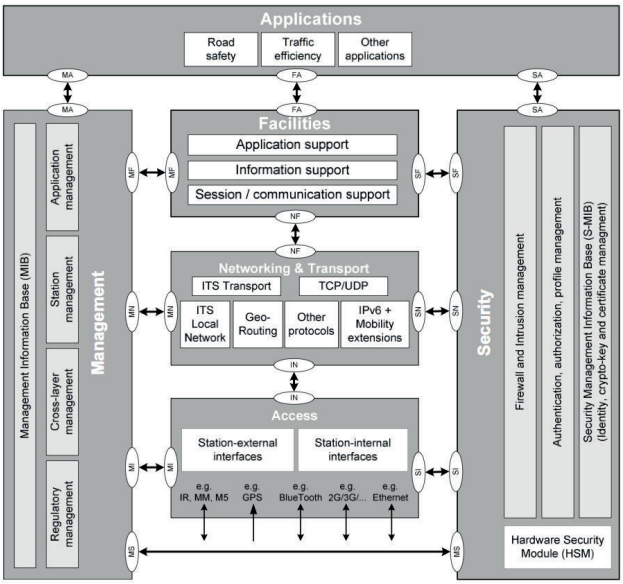

Figure 1: ETSI architecture of an OBU [10]

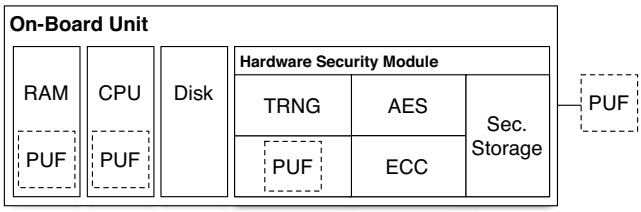

Figure 2: Simplified hardware architecture of an OBU

The formulation of abstract properties of PUF types as well as the development of PUF constructions are still a matter of active research [36]. In this paper we use the terminology proposed by Rührmair et al. [39] and refer to Strong PUFs[1] and Weak PUFs[2]. To the best of our knowledge, no research has investigated the applicability of PUFs for storage of large numbers of private keys (or keypairs) as required by the V2X pseudonym scenario.

3.2 On-Board Unit Architecture

Figure 1 shows the current ETSI reference architecture of an On-Board Unit (named "*ITS Station*" in the standard). It shows the different layers and particularly the security layer. One can notice the Hardware Security Module (HSM) within. As Figure 1 is an abstract view of an OBU, and thus, does not represent the hardware, Figure 2 shows a simplified hardware architecture. An OBU includes CPU, host

[1]Labeled as "minimum readout time" PUF (MRT-PUF) [36]
[2]Also known as Physical Obfuscated Keys (POKs)

memory (RAM), regular storage, and an HSM. For simplicity, we represent an HSM that only includes a true random number generator (TRNG), cryptographic primitives (AES, ECC), secure storage, and a PUF. However, one should notice that the PUF could be outside of the HSM (represented in dashed line in Figure 2). Indeed, the PUF could be fully integrated in the CPU, GPU or RAM [26], but also attached to the OBU as an external device. In the remainder of this paper, we consider the PUF as an external device as we compare against classic secure storage solutions (e.g. smart card, secure token), which are mostly externally attached to the OBU. A consequence of being outside the HSM is the lack of a secure computation environment. An attacker (described in Section 3.3) could then access to the memory to steal key material. However, this drawback is limited by the limited lifetime of the pseudonyms (i.e. certificate). We further discuss the issue of secure computation in Section 6.2.

3.3 Attacker Model

With respect to secure storage we consider attackers who want to access the content that is placed in the secure storage container. In our context, the aim of the attacker is to copy the private key material used as pseudonym of a vehicle. We differentiate between two attacker goals: An attacker might try to get access to the private keys for the currently used pseudonym or the attacker might aim to access all private keys for all pseudonyms provisioned in the OBU.

An attack against the OBU can be performed by injecting a payload into the system, which would trigger malicious actions. Since the OBU does not provide a user interface, such a payload needs to be injected into the system remotely. OBUs offer a number of opportunities to an attacker to input data into the system remotely. Most notably the networking and communication applications in the OBU are processing data from external sources, which might be controlled by an attacker. Exploitation of security holes in these applications can lead to different levels of access to the contents of the OBU:

1. Access to filesystem data
2. Access to application memory
3. Access to hardware devices and code execution

We consider attackers with an escalating set of capabilities to evaluate the level of protection offered by the different proposed techniques. Access to filesystem data serves as a baseline scenario to illustrate that the basic secure storage mechanisms work. An attacker should never be able to access key material based on filesystem access. The second level of access represents more severe information disclosure attacks. In a scenario without secure computation this will allow an attacker to extract key material that is currently in use. A third type of attacker has the ability to execute arbitrary code on the OBU, and thus, is able to arbitrarily interact with any device attached to the OBU. For an external device, such an attacker is indistinguishable from a regular host application. Nevertheless, we consider this type of attacker as the most powerful type of attacker, because she has full control of the OBU.

In this paper we do not investigate hardware attacks against the secure storage. The intrinsic tamper-resistance of PUFs is assumed to protect against this kind of attacker. We assume equally that the tamper-proof enclosure of classic secure storage solutions is effective.

4. CLASSIC SECURE STORAGE

In this section we propose ways to implement efficient secure storage of large numbers of private keys for use in secure pseudonymous communication. We differentiate between regular storage and secure storage requirements for keys and related support data. The proposed solutions have different space requirements to store and protect these data, which will be our main metric to compare the efficiency of the proposed methods. As a baseline, we assume the availability of classic external secure storage, for example in the form of a physical *smart card* or as part of a dedicated secure *storage token* on a USB device. Our goal is to minimize the usage of this resource or eliminate the use of this resource entirely.

4.1 Individual key storage

The canonical way to handle secure storage is to assume the presence of a dedicated device, which is isolated from the host. The security attributes of this solution are derived from the fact that the memory on this type of device is only accessible through a well-defined security API. No other way should exist to access the data, neither in software nor in hardware. The protection against hardware access is usually achieved through protective tamper-proof enclosures or self-destructive coating. The details of the hardware and the communication protocol as well as options to perform secure computation on the device are out of the scope of this paper.

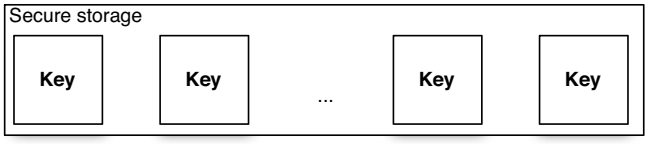

Figure 3: All keys in secure storage

Figure 3 illustrates the fact that all n keys need to be stored in the secure external device. The limiting factor of this solution is the raw amount of data that needs to be stored in this scheme. As introduced in Section 1, it is expected that secure pseudonymous communication in vehicular networks will require multiple megabytes of private keys. The key management and the amount of secured data storage increase the cost of such a solution as the number of pseudonyms grows.

4.2 Encrypted storage

Storing private keys in encrypted form in regular storage, e.g. in a file or database, is a common solution found in password management software for consumers (see Section 2.3). This kind of solution is usually tied to a master password and a password-based key derivation function to decrypt the data structure. For non-interactive use, we can adapt this solution to use a master key stored inside a secure storage device to encrypt and decrypt the private keys as needed. Figure 4 illustrates this method. Using a master key with sufficient entropy in a secure data store allows us to avoid

key stretching techniques [21] that are typically employed in password based key derivation functions like PBKDF2 [18], bcrypt [37], or scrypt [34].

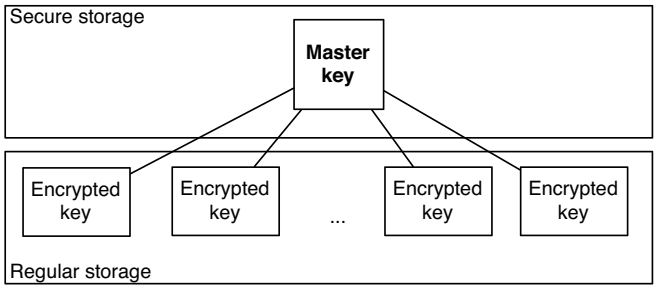

Figure 4: Keys retrieved from encrypted file in regular storage using a securely stored master key

The advantage of this method compared to a classic secure storage solution (Section 4.1) is that only one master secret is required to be stored securely. This master secret will subsequently unlock any number of additional private keys, which can be stored in encrypted form in regular unsecured memory. Conversely, the disadvantage is of course that now an attacker only requires this master key and the encrypted–but not securely stored–data structure of private keys to not just compromise one private key, but all private keys stored in this data structure.

4.3 Key derivation

Taking the concept of using a master secret even further, we use a key derivation function to derive secret keys from the master secret. A practical implementation of this idea uses a keyed pseudo-random function to derive a sequence of bits from a single master key (or seed). These bits can be used as a secret key for symmetric cryptography, but also as a deterministic source of random bits in the generation process of an asymmetric ECDSA key pair [13]. Figure 5 illustrates this abstract process. Well known constructions of such key derivation functions include KDF2 [16, 17, 9], HKDF [24, 25, 4], and the set of deterministic random bit generators (without reseeding) recommended by NIST [2].

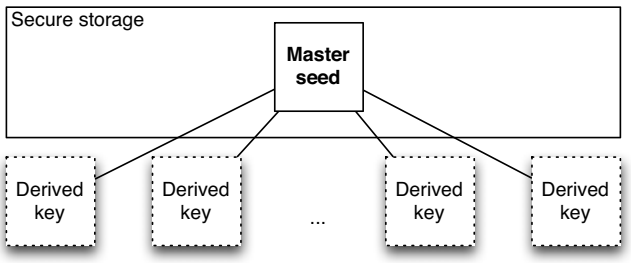

Figure 5: Keys regenerated through a key derivation function using a securely stored master key

An additional advantage of using key derivation functions is the reduction of the communication overhead. Indeed, if the CA generates and stores the master seed for the vehicles, it is no longer necessary to submit the key pair through a secure communication channel to the vehicle. It is enough to transfer only the certificates, which needs to include context information, to allow the vehicle to derivate the matching key pair independently. These information do not even require protection, enabling the use of unauthenticated broadcast channels or public certificate servers for the delivery of new pseudonym certificates.

5. PUF-BASED SECURE STORAGE

In this section we propose secure key storage solutions which do not rely on any classic external secure storage, but instead, use Physical Unclonable Functions (PUFs) to achieve the desired security. As introduced in Section 3.1, we consider two types of PUFs: Strong PUFs and Weak PUFs.

5.1 Strong PUF-based secure storage

Ongoing research on applications of PUFs for key generation and regeneration is focusing on the fuzzy extraction algorithm. From an application perspective in the vehicular communication context, we observe that we need to securely store large numbers of secret keys. Our proposal, which is summarized in Figure 6, requires the use of a Strong PUF [39] that fulfills the following requirements:

1. It must be impossible to physically clone the PUF.
2. A complete determination/measurement of all challenge-response pairs (CRPs) within a limited time frame (such as several days or even weeks) must be impossible.
3. It must be practically impossible to numerically predict the response to a randomly selected challenge, even if many other CRPs are known.

These requirements were setup by Rührmair et al. with scenarios in mind that require a large number of interactive challenge-response cycles, e.g., for remote authentication. Attackers could, e.g., send specific challenges to the PUF, record the responses, and then try to perform a so called "model building attack" [39]. For our usage of PUFs for pseudonym storage, an attacker will not be able to directly query the PUF and see the responses. Only the CA is supposed to be able to communicate with the OBU, and PUF responses will only be used to derive key pairs from it. This effectively removes the unconditional need for requirement 2, although for cost effectiveness of this solution it is still desirable to demand a large space of challenge-response pairs (CRPs). In Section 5.2, we propose an alternative solution that can tolerate the availability of only small amount of CRPs per PUF (Weak PUF).

The idea that we pursue in this proposal is to derive key material from PUF responses. The use of a Strong PUF implies that we have a large space of challenge-response pairs, which enables us to derive large numbers of keys. As in the solution based on KDF, we use deterministic random bits as a source of entropy in the key generation process of asymmetric ECDSA key pairs [13].

The amount of input data required to generate a stable amount of responses is highly dependent on the attributes

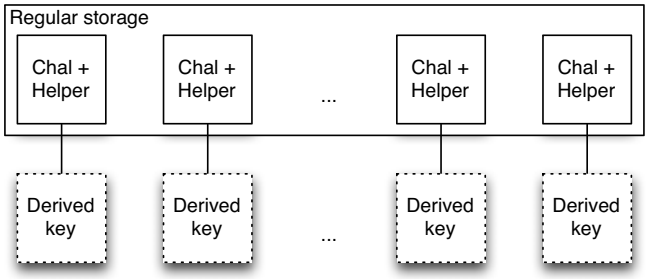

Figure 6: Keys reconstructed securely from a strong PUF using regularly stored challenges and helper data

of a concrete PUF construction. In general we require a set of chosen challenges and a set of helper data, which is generated by the fuzzy extractor during the initial key generation process. Depending on the type of PUF construction, a total amount of n challenges c_n of x bits length is required to generate m bits of output. These m bits of output then need to be stabilized using a fuzzy extractor (see Section 3.1). In the initial key generation process the fuzzy extractor will generate helper data. In subsequent calls to the PUF, this helper data is instead used by the fuzzy extractor to reconstruct the same stable response. In both cases, the fuzzy extractor will consume a percentage of the data for entropy compression and error correction. The factor of the data reduction r as well as the length y of the helper data W depends on the type and configuration of the fuzzy extractor. The configuration needs to be calibrated based on the expected *error probability* and *entropy quality* of a given PUF construction.

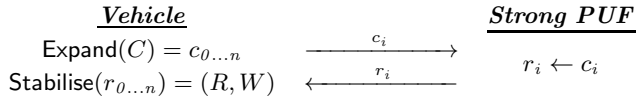

Figure 7: An initial challenge (C) gets expanded into n challenges (c_i), which generate responses (r_i) in the PUF. The vehicle combines these into a final response (R) and helper data (W).

For an overall amount of stable response bits z, we can calculate the number of required challenges as $n = \frac{z}{m \cdot r}$. To enable reconstruction of stable responses, we would need to store the n challenges of size x and the helper data W of length y. Regarding the choice of challenges, we note that to ensure the independence of output bits we need to avoid repetitions of challenges. A simple increment function allows us to easily expand multiple challenges from an initial challenge, while avoiding collisions and covering the whole space of possible challenges optimally. Under the assumption that the number of challenges to expand is implicitly known for each reconstruction of a response, this makes it possible to only store the starting challenge and derive all following challenges. Thus, to enable reconstruction of fixed size stable responses, we need to store only the starting challenge of size x and the helper data W. Once all possible challenges are exhausted the PUF should not be reused[3]. Requirement 3 of the Strong PUF definition, as well as the attributes of the fuzzy extractor, must ensure that even just a one bit difference between challenges guarantees a fully independent response.

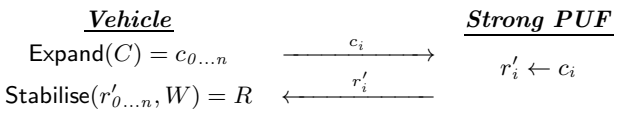

Figure 8: Regeneration of responses is analogous to the initial provisioning, except the previously generated helper data (W) is now utilized by the Stabilise() function to stabilize the response.

The details of the ECDSA key pair generation process are specified in [13]. For example a fixed amount of 320 random bits are required to deterministically build a key pair of 256 bits. Thus, we assume a need of $z = 320$ bits of stable entropy from the PUF to be able to generate a 256 bit ECDSA key pair.

Once the vehicle has constructed its key pair as outlined above, it can then build and submit a certificate signing request (the public key) to the CA through a authenticated and integrity protected channel to trigger the certification process. The CA subsequently returns a signed certificate, which completes the provisioning process of a new pseudonym.

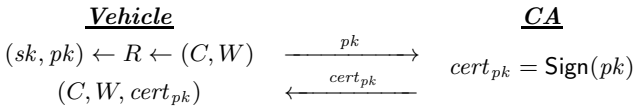

Figure 9: The vehicle generates an asymmetric key pair from a challenge C and helper data W. The CA creates a certificate for the public key pk, which is stored in the vehicle with C and W.

This method of secure key generation and key reconstruction from PUFs completely avoids any need for classic secure storage. The starting challenge and helper data can be stored in regular storage space. The security of the key material is fully guaranteed by the need to have access to the related PUF device with its intrinsic tamper resistant attributes.

5.2 Weak PUF-based key derivation

A Weak PUF deviates from the definition of Strong PUF by allowing just one fixed CRP per PUF. It can be considered as a PUF that has a fixed built-in challenge and whenever queried provides the same response. This leads to the violation of requirements 2 and 3 of the Strong PUF definition as described above. Nevertheless, even if the Weak PUF has a capacity of one single CRP, this CPR will have a useful amount of entropy. Assuming that the size of the response provides sufficient entropy for a master secret as described in Section 4.3, we can apply the same technique here.

[3]Reconfigurable PUFs have been proposed as a desirable extension [20]

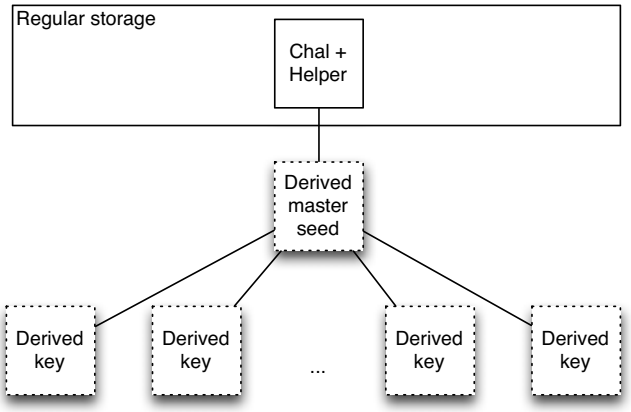

Figure 10: A master key gets reconstructed securely from a weak PUF using regularly stored challenges and helper data and is then used to regenerate derived keys.

An illustration of the two stage key derivation process is shown in Figure 10. First, a master key is derived from the response of a Weak PUF. Then, this master key is used as a seed to derive the key material for multiple pseudonyms. For instance, the PUF response could be used as the "input keying material" for the *Extract* function of HKDF [25].

6. DISCUSSION

The presented solutions for the secure storage of key material for pseudonyms employ Key Derivation Functions (KDF) and Physical Unclonable Functions (PUF) to achieve multiple levels of efficiency improvements. The two major aspects for the evaluation of the solutions are the *storage requirements* and the *security properties* with respect to attackers with different capabilities.

In Table 1 we summarize the storage requirements of the proposed solutions for the secure storage of k keys. Based on the assumption of storing $k = 105120$ keys, the baseline classic secure storage scenario would require approximately 3.2 Mbytes of secure storage space. An encrypted data structure, as described in Section 4.2, would allow to drastically reduce the amount of secure storage. In this scenario, the full 3.2 Mbytes of encrypted key material still has to be stored, but it can be stored in regular memory.
The use of a key derivation function removes this requirement of regular storage by relying purely on a master seed value, which is used to generate key material on-the-fly.
The solution based on the application of a Strong PUF does not require any classic secure storage device at all. Instead, it is possible to rely solely on the intrinsic security of the PUF construction. However, the amount of regular storage space required to regenerate keys is larger than the raw amount of private keys. This is due to the need for helper data, which is required to stabilize the readings of responses from the noisy hardware constructions of PUFs. The exact amount of required helper data and the size of challenges are highly dependent on the attributes of a given PUF and also on algorithmic choices of the fuzzy extractor.
Finally, we see that a combination of PUF and KDF techniques even allows us to present a solution that technically does not require any secure or regular storage at all. The Weak PUF using just one challenge-response pair, returns its response without any explicit challenge, simply by virtue of being powered on.

The second criteria to compare the proposed solutions is the resilience against attackers with different levels of capabilities (see Section 3.3). Table 2 gives an overview of the security properties. We see that all solutions guarantee the basic requirement of denying any access to the key material to an attacker who has access to the regular unsecured filesystem.
As described in Section 3.3, the next level of attacker capability grants the attacker read-only access to arbitrary regions of OBU memory. The attacker might have found an exploitable bug in the software and injects malicious code to extract valuable data. We see weaknesses in three of the proposed solutions, due to the fact that these rely on a single piece of master secret to derive key material. This master secret (a master key or a master seed) has to be extracted from a classic secure storage device or from a PUF and is identical for all keys that are derived by the system. An attacker with the capability to observe the address space of the application can potentially copy this master secret during the derivation process of any key. The attacker can then derive all possible keys based on this master secret. Only the pure classic secure storage solution and the Strong PUF based solution are not affected by this issue, because these solutions derive all keys independently.
The final model grants the attacker full control over the host, which implies code execution privileges and direct access to the device. Generally, there is no way to protect the information against access by such a powerful attacker, because the storage device cannot see a difference between normal usage and usage by such a powerful attacker. One last option to offer a mitigation against malicious use could be a rate limitation mechanism, which limits the number of requests over time. For the use case of pseudonymous communication in vehicular communication it could be sufficient to only return one key per minute. Such a feature represents a viable security benefit, because the attacker can effectively only make use of the attacked device while it is online. The classic secure storage solution, as well as the Strong PUF-based solution, could reasonably offer such a feature.

6.1 Limitations of KDFs and PUFs

In the previous section, the comparison of the security properties listed in Table 2 shows that the existence of a single master secret, as it is the case in the KDF-based solutions, represents a disadvantage under certain attacker models. Another issue to consider is the limitation of the number of keys to derive from one single master key. It is advisable to rekey the system after a certain amount of keys was derived. The rekeying interval depends on the construction of the underlying algorithm used in the KDF. This also highlights the abstract disadvantage of having to rely on additional cryptographic algorithms compared to the solutions that access keys without intermediary derivation steps. More exposure to cryptographically strong algorithms naturally implies more risk of being affected by a discovery of a weakness in such algorithms.

Table 1: Storage size overview for k keys

	Secure Storage	Regular Storage	Comments
Classic secure storage, Section 4.1	k private keys	–	ECDSA private key size ≈ 256 bit
Encrypted storage, Section 4.2	1 master key	k encrypted private keys	Master symmetric key size ≈ 128 bit
Key derivation, Section 4.3	1 master seed	–	Master seed size ≈ 320 bit
Strong PUF-based secure storage, Section 5.1	–	k challenges, k helper data	Size of challenge and helper data is highly dependent on PUF construction. Chen et al. recommend challenge sizes of 64 or 128 bits for a BR-PUF [6].
Weak PUF-based key derivation, Section 5.2	–	helper data	A Weak PUF does not necessarily require a challenge. Maes et al. [30] list $y = 2052$ bits of helper data for a response of 128 bits from an RO-PUF

Table 2: Key stealing protection under different attacker capabilities

	Filesystem access	Memory access	Full control of OBU
Classic secure storage, Section 4.1	safe	current key accessible	rate limitation possible
Encrypted storage, Section 4.2	safe	all keys accessible	all keys accessible
Key derivation, Section 4.3	safe	all keys accessible	all keys accessible
Strong PUF-based secure storage, Section 5.1	safe	current key accessible	rate limitation possible
Weak PUF-based key derivation, Section 5.2	safe	all keys accessible	all keys accessible

Similar concerns are valid for PUFs, where the fuzzy extraction process is comparable to a key derivation process. The complexity of these processes might enlarge the exposure to bugs and weaknesses. Moreover, there are fundamental capacity limits (i.e. challenge-response pair space) that might impede practical deployments. Since PUFs are intrinsically bound to hardware, it might be impossible to reuse (rekey) a PUF after the capacity limit is reached. This is particularly problematic for Weak PUFs with only one or a very limited number of challenge-response pairs. Controlled PUFs and Reconfigurable PUFs [20] have been proposed as solutions for this problem, but the feasibility of such constructions is hard to evaluate. A controlled PUF would be particularly desirable in the context of secure storage for the possibility to effectively implement rate limitation in hardware.

While it is not an issue for pseudonyms storage in vehicular network, we acknowledge that the speed of accessing a PUF can be a limiting factor. The secure key reconstruction from PUFs incurs a considerable amount of computational overhead for the fuzzy extraction of responses. According to [30] the execution time is in the order of magnitude of several milliseconds for an RO-PUF design. Additionally, the challenge C and helper data W, which need to be stored for the regeneration of a stable response, are significantly larger than the plain private key. While we propose an expansion function to avoid storing all challenges c_n, the helper data can easily add up to several kilobytes in order to generate stable response data [30].

Another limitation of using PUFs for key generation and key storage is that PUFs are effectively read-only devices. Therefore, it is necessary for vehicles to create key pairs locally, using the response of a PUF challenge as a controlled source of entropy.

We summarize the limitations of PUFs as follows:

1. Read-only data store
2. Limited capacity
3. Readout time
4. Faith in fuzzy extractor algorithms
5. Need to store helper data

These limitation pose restrictions on the realm of possible applications for PUF-based secure storage. PUF-based solutions are consequently not suitable as a direct universal replacement for all applications of classic secure storage. Nevertheless, when these limitations are met, the use of PUF-based solutions is a secure and efficient option to replace classic secure storage.

6.2 PUF integrated within an HSM

As shown in Figure 2, the PUF could be inside an HSM. Then, our schemes would benefit from this secure computation environment. Indeed, an HSM commonly provides secure memory, secure storage, and secure cryptographic primitives. This solution ensures that the key is generated and used at the same place, and never leaves the HSM. In this case, one can notice that integrating the PUF inside the HSM will prevent all the key stealing attacks listed in Table 2.

However, an attacker with full access could still use the HSM to perform malicious actions such as signing forged message. Moreover, the PUF limitations still hold even within an HSM. For instance, the limited capacity of the challenge space triggers the question about what would happen when a CRP space is depleted. As no Reconfigurable PUF exists yet, replacing the HSM would incur a considerable cost.

Finally, we conclude that if secure computation is assumed,

then the cost benefit advantage of PUF is questionable. We note that the encrypted storage model (Section 4.2) would not suffer from any limitation of the PUF-based solutions while offering a better tradeoff between secure storage and regular storage. According to Table 1, *encrypted storage* needs 1 private key and k cipher texts, while *PUF-based* approaches require no private key but k challenges and k helper data. One should notice that, in terms of size, the cipher text is significantly smaller than the set of challenges and helper data.

7. CONCLUSION AND FUTURE WORK

Security and privacy of vehicular communication are mandatory to ensure a successful deployment and user acceptance of cooperative Intelligent Transportation System. The current set of V2X standards foresees the use of asymmetric cryptography, digital signatures, and certificates to authenticate users. To prevent tracking and privacy leakage, vehicles frequently switch between short-term pseudonyms to provide anonymity and unlinkability. As permanent–or even frequent–connection to the PKI to retrieve new pseudonyms cannot be guaranteed, a common solution is to store enough pseudonyms for one year or longer in secure storage. However, secure storage of large amount of key material is expensive if done in secure memory of a hardware security module.

In this paper, we propose to use encryption and key derivation functions to reduce the need for secure storage. Our comparison shows that the use of these techniques are effective at reducing the requirements for secure storage at the cost of reduced protection against attackers with access to host memory. We alternatively propose to use Physical Unclonable Functions (PUFs) to eliminate the need for classic secure storage entirely. Our analysis shows that PUFs can effectively replace classic secure storage if an application can operate under the limitations of a given PUF. The use-case of secure pseudonymous communication in vehicular networks is generally compatible with these limitations.

The attractiveness of PUF-based solutions is a result of potential cost savings due to the use of PUF constructions compared to more expensive secure storage. PUFs are envisioned to be cheap enough for inclusion in mass produced RFID tags or might already exist in common hardware. This represents a considerable cost-benefit advantage. Once the availability of hardware implementations increases, we expect PUF-based solutions, such as the storage solutions presented in this paper, to see widespread use in practical applications.

As future work, we point out that detailed assumptions about the behavior of PUFs are often hard to verify. In this paper we require two properties about PUFs that allow us to implement optimizations and make assumptions about the security of the overall system:

1. A one bit difference between two challenges is enough to guarantee completely independent responses. Knowledge of related (not randomly selected) challenges does not affect the unpredictability of responses.
2. Knowledge of helper data does not reveal any information about the expected response from a PUF.

These attributes are implied by the Strong PUF requirement 2 and by the fuzzy extraction algorithm goals. But usually no explicitly guarantees of these attributes are given in the design documents of concrete PUF constructions.

Applications of secure storage in vehicular OBUs often involve full Hardware Security Modules (HSM) to provide secure computation in addition to secure storage. Rate limitation and a limited lifetime of certificates do allow operation without secure computation. It remains an open question, if a PUF-based secure storage solution can be augmented to offer secure computation, while retaining a cost-benefit advantage over classic implementations in an HSMs.

Development of PUF constructions is a very active area of research and we hope that new developments might remove some of the aforementioned limitations.

8. ACKNOWLEDGEMENTS

The authors would like to thank Dominik Merli and Christoph Bösch for their helpful comments. The research leading to these results has received funding from the European Union's Seventh Framework Programme project PRESERVE under grant agreement n°269994.

9. REFERENCES

[1] L. Apvrille, R. El Khayari, O. Henniger, Y. Roudier, H. Schweppe, H. Seudié, B. Weyl, and M. Wolf. Secure automotive on-board electronics network architecture. *World Automotive Congress (FISITA '10)*, May 2010.

[2] E. B. Barker and J. M. Kelsey. *Recommendation for random number generation using deterministic random bit generators (revised)*. US Department of Commerce, Technology Administration, National Institute of Standards and Technology, Computer Security Division, Information Technology Laboratory, 2007.

[3] G. Calandriello, P. Papadimitratos, J.-P. Hubaux, and A. Lioy. Efficient and robust pseudonymous authentication in vanet. *4th ACM international workshop on Vehicular ad hoc networks (VANET '07)*, pages 19–28, 2007.

[4] L. Chen. SP 800-56C. recommendation for key derivation through extraction-then-expansion. Technical report, Gaithersburg, MD, United States, 2011.

[5] N. Chen, M. Gerla, D. Huang, and X. Hong. Secure, selective group broadcast in vehicular networks using dynamic attribute based encryption. *9th IFIP Annual Mediterranean Ad Hoc Networking Workshop (Med-Hoc-Net '10)*, pages 1–8, 2010.

[6] Q. Chen, G. Csaba, P. Lugli, U. Schlichtmann, and U. Rührmair. The bistable ring puf: A new architecture for strong physical unclonable functions. *IEEE International Symposium on Hardware-Oriented Security and Trust (HOST '11)*, pages 134–141, 2011.

[7] T. C. S. de Souza, J. E. Martina, and R. F. Custódio. Audit and backup procedures for hardware security modules. *7th Symposium on Identity and Trust on the Internet (IDtrust '08)*, pages 89–97, 2008.

[8] Y. Dodis, M. Reyzin, and A. Smith. Fuzzy extractors: How to generate strong keys from biometrics and other noisy data. In *Advances in Cryptology –*

EUROCRYPT 2004, volume 3027 of *LNCS*, pages 523–540, 2004.

[9] B. Elaine, J. Don, and S. Miles. SP 800-56A. recommendation for pair-wise key establishment schemes using discrete logarithm cryptography. Technical report, Gaithersburg, MD, United States, 2007.

[10] ETSI TC ITS. ETSI TS 102 731 v1.1.1 - intelligent transport systems (ITS); security; security services and architecture. Standard, 2010.

[11] ETSI TC ITS. ETSI TS 102 941 v1.1.1 - intelligent transport systems (ITS); security; trust and privacy management. Standard, 2012.

[12] ETSI TC ITS. ETSI TS 103 097 v1.1.1 - intelligent transport systems (ITS); security; security header and certificate formats. Standard, 2013.

[13] Federal Information Processing Standards. *Digital Signature Standard (DSS) - FIPS 186-3*, June 2009.

[14] T. Foundation. Truecrypt - free open-source on-the-fly encryption, 2013. Retrieved July 10, 2013 from http://www.truecrypt.org/.

[15] D. Garcia, A. Waite, R. Walsh, B. Sheppard, L. Frank, and D. Jeffers. Certificate management entities for connected vehicle environment. public workshop read-ahead document. Technical report FHWA-JPO-12-038, Research and Innovative Technology Administration, May 2012.

[16] IEEE. IEEE standard specifications for public-key cryptography- amendment 1: Additional techniques. *IEEE Std 1363a-2004 (Amendment to IEEE Std 1363-2000)*, pages 1–159, 2004.

[17] ISO/IEC. Information technology - security techniques - encryption algorithms - part 2: Asymmetric ciphers. *ISO/IEC 18033-2*, 2006.

[18] B. Kaliski. RFC 2898: Pkcs# 5: Password-based cryptography specification version 2.0. *IETF, September*, 2000.

[19] F. Kargl, P. Papadimitratos, L. Buttyan, M. Muter, E. Schoch, B. Wiedersheim, T.-V. Thong, G. Calandriello, A. Held, A. Kung, and J.-P. Hubaux. Secure vehicular communication systems: implementation, performance, and research challenges. *IEEE Communications Magazine*, 46(11):110–118, November 2008.

[20] S. Katzenbeisser, Ü. Kocabaş, V. van der Leest, A.-R. Sadeghi, G.-J. Schrijen, and C. Wachsmann. Recyclable PUFs: Logically reconfigurable PUFs. *Journal of Cryptographic Engineering*, 1(3):177–186, 2011.

[21] J. Kelsey, B. Schneier, C. Hall, and D. Wagner. Secure applications of low-entropy keys. In *Information Security*, pages 121–134. Springer, 1998.

[22] B. H. Kim, K. Y. Choi, J. H. Lee, and D. H. Lee. Anonymous and traceable communication using tamper-proof device for vehicular ad hoc networks. *International Conference on Convergence Information Technology*, pages 681–686, 2007.

[23] W. Koch. Gnupg - the gnu privacy guard, 2013. Retrieved July 10, 2013 from http://gnupg.org/.

[24] H. Krawczyk. Cryptographic extraction and key derivation: The HKDF scheme. In *Advances in Cryptology–CRYPTO 2010*, pages 631–648. Springer, 2010.

[25] H. Krawczyk and P. Eronen. HMAC-based Extract-and-Expand Key Derivation Function (HKDF). RFC 5869, May 2010.

[26] T. Lange. PUFFIN - the physically unclonable functions found in standard pc components project, 2013. Retrieved July 10, 2013 from http://puffin.eu.org/.

[27] T. Leinmüller, L. Buttyan, J.-P. Hubaux, F. Kargl, R. Kroh, P. Papadimitratos, M. Raya, and E. Schoch. Sevecom - secure vehicle communication. *IST Mobile and Wireless Communication Summit*, pages 1–5, 2006.

[28] J.-P. M. G. Linnartz and P. Tuyls. New Shielding Functions to Enhance Privacy and Prevent Misuse of Biometric Templates. *Audio-and Video-Based Biometrie Person Authentication (AVBPA '03)*, 2688:393–402, 2003.

[29] A. Lysyanskaya, R. L. Rivest, A. Sahai, and S. Wolf. Pseudonym systems. *6th Annual International Workshop on Selected Areas in Cryptography (SAC '99)*, pages 184–199, 1999.

[30] R. Maes, A. Herrewege, and I. Verbauwhede. PUFKY: A fully functional puf-based cryptographic key generator. *Cryptographic Hardware and Embedded Systems (CHES '12)*, pages 302–319, 2012.

[31] K. Moerman, T. van Roermund, and M. Knezevic. A realistic approach to message verification in car-to-car communication. *19th ITS World Congress*, 2012.

[32] R. Pappu. *Physical One-Way Functions*. PhD thesis, MIT, 2001.

[33] R. Pappu, B. Recht, J. Taylor, and N. Gershenfeld. Physical One-Way Functions. *Science*, 297:2026–2030, 2002.

[34] C. Percival. Stronger key derivation via sequential memory-hard functions. *The Technical BSD Conference (BSDCan '09)*, May 2009.

[35] J. Petit, C. T. Bösch, M. P. Feiri, and F. Kargl. On the potential of puf for pseudonym generation in vehicular networks. *4th IEEE Vehicular Networking Conference (VNC '12)*, pages 94–100, November 2012.

[36] R. Plaga and F. Koob. A formal definition and a new security mechanism of physical unclonable functions. In *Measurement, Modelling, and Evaluation of Computing Systems and Dependability and Fault Tolerance*, pages 288–301. Springer, 2012.

[37] N. Provos and D. Mazieres. A future-adaptable password scheme. *USENIX Annual Technical Conference, FREENIX Track*, pages 81–91, 1999.

[38] D. Reichl. Keepass password safe, 2013. Retrieved July 10, 2013 from http://keepass.info.

[39] U. Rührmair, F. Sehnke, J. Sölter, G. Dror, S. Devadas, and J. Schmidhuber. Modeling attacks on physical unclonable functions. *17th ACM conference on Computer and communications security (CCS '10)*, pages 237–249, 2010.

[40] B. Schneier. Password safe - the security of twofish in a password database, 2013. Retrieved July 10, 2013 from http://www.schneier.com/passsafe.html.

[41] B. Wiedersheim, Z. Ma, F. Kargl, and

P. Papadimitratos. Privacy in inter-vehicular networks: Why simple pseudonym change is not enough. *7th International Conference on Wireless On-demand Network Systems and Services (WONS '10)*, 2010.

[42] M. Wolf, A. Weimerskirch, and T. Wollinger. State of the art: Embedding security in vehicles. *EURASIP Journal on Embedded Systems*, 2007, 2007.

[43] L.-Y. Yeh, Y.-C. Chen, and J.-L. Huang. ABACS: An attribute-based access control system for emergency services over vehicular ad hoc networks. *IEEE Journal on Selected Areas in Communications*, 29(3):630–643, 2011.

[44] B. Yu, C.-Z. Xu, and B. Xiao. Detecting sybil attacks in vanets. *Journal of Parallel and Distributed Computing*, 73(6):746–756, 2013.