Physical Unclonable Functions and Their Applications to Vehicle System Security (Full Paper)

Muhammad Asim, Jorge Guajardo, Sandeep S. Kumar, and Pim Tuyls

Philips Research Europe, Eindhoven, The Netherlands
{muhammad.asim,jorge.guajardo,sandeep.kumar,pim.tuyls}@philips.com

Abstract. In recent years, there has been a tremendous increase in the usage of IT based systems in vehicles, with predictions that in the near future, more than 90% of innovations in the automotive sector will be centered on IT software and hardware. However, innovation also means that IP is created, which is valuable to third (potentially) untrusted and malicious parties. In particular, automobiles are already suffering from security issues, such as illegal copying of software intellectual property (IP), counterfeiting of electronic components, illegal tampering with digital data inside the electronic control units (ECUs), etc. Recently Physical Unclonable Functions (PUFs) attracted significant interest for numerous applications such as protection of software and hardware IPs, secure key storages and components identification, to name a few. In this paper, we describe how PUFs can be used for secure key storage, component identification, and IP protection in vehicle applications. In addition, the suitability of PUFs for vehicle insurance applications is explained.

1 Introduction

Today we are used to observing improvements in processors every 18 months, 50% price decreases in less than a year periods, and new technological innovations every day of our lives. As expected, every aspect of our lives is subject to this fast moving and changing society, including the automobile. Today's automobile are complex systems including over 70 processors and over hundred megabytes of program code [38]. Tomorrow's automobiles will include connection to the Internet, real-time road condition information, and many other innovation that we can not yet imagine. However, it is our belief that many of these innovations will only be successful if integrated into a framework which takes into account both security and safety concerns. In fact, it is already clear that counterfeiting of parts destined for automobiles and planes is a lucrative business as well as a dangerous one for the end consumer [15]. Notice also that counterfeiting is not limited to the automobile industry. Rather, it is a much larger problem as the following examples will demonstrate:

- It is estimated that global economic damage across all industries due to the counterfeiting of goods surpasses the 600 billion per year mark [27]. In India, 15% of fast-moving consumer goods and 38% of auto parts are counterfeited.
- In addition to the brand damage that a particular company may suffer, counterfeited products have impact on our safety as well. For example, counterfeited spare parts of planes have caused planes to crash [15] and above all counterfeiting in the medicine is a direct threat to the existence of humanity.
- Counterfeiting in the software sector is a real threat as digital content can be easily copied. Notice that the costs for software and electronics are estimated to approach the 50% margin in car manufacturing by 2015 [29]. Perhaps more importantly, there are estimates that already today more than 90% of all vehicle innovations are centered on IT software and hardware [26].

Thus, it is no surprise that a lot of work has gone into developing solutions to prevent security breaks in automobile systems [20, 36, 1, 37, 4, 30]. Moreover, and as a result of the need for security, future automobile applications will make use of standard cryptographic primitives and security solutions. Observe that a widely accepted security principle is that of basing the overall security of the system on the secrecy of the key. Thus, protecting keys from compromise seems of the utmost importance. In this paper, we suggest that a cheap, secure and efficient manner to achieve the above goal is via Physical Unclonable Functions (PUFs). We show in the paper that PUFs can also be used as unclonable identifiers and in combination with private-key and public-key primitives to safeguard intellectual destined for the automobile. In addition, we suggest that PUFs would be advantageous for insurance purposes as well.

The remainder of this contribution is organized as follows. Section 2 provides a brief introduction to error correcting codes and universal hash functions, which are basic building blocks for helper data algorithms. The definitions of PUFs, their properties, the assumptions made on their behavior and implementation, helper data algorithm used to cope with the noisy nature of physical processes (such as PUFs) are introduced in Sect. 3. In Sect. 4, we provide an summary of the state-of-the-art solutions for security in cars, heavily based on [38]. We also described in some detail the work on component identification of [36], on which we based our some of our discussion on uses of PUFs in automobiles. Section 5 describes four applications of PUFs which we consider relevant to the automobile setting: secure key storage, component identification, protection of IP to be used in the car both as part of the car (manufacturer IP) and for the end user, and finally we describe how PUFs could enhance the work of insurance companies when verifying the authenticity of parts and their role in accidents.

2 Preliminaries

In our discussion of physical unclonable functions and their use in anti-counterfeiting applications we will assume familiarity with standard cryptographic blocks such as symmetric-key primitives (e.g. the AES, DES, triple-DES), hash functions (e.g. MD5, SHA-1, SHA-2) and public-key based primitives (RSA, elliptic curves). We will also make use of error correcting codes and universal hash functions. In order to make the treatment self-contained, we will provide a brief introduction to the latter subjects in what follows. For an introduction to standard crytpto primitives in the context of automotive security we refer for example to [38].

2.1 Error Correcting Codes

We will begin by informally defining block codes. A block code is essentially a set and a pair of algorithms. The first one adds redundancy to a message (via an encoding algorithm) so that upon receiving the encoded (noisy) message, this can be decoded with minimal errors using the decoding algorithm. One important characteristic that differentiates block codes from other codes (e.g. convolutional codes) is that they are fixed length. In particular, a block code encodes k-digit messages into n-digit codewords. Digits corresponds to the symbols from a set or alphabet, which often is a finite field and in this paper will be the binary alphabet i.e. $\{0,1\} \in \mathbb{F}_2$.

More formally, a binary linear code $\mathcal{C}$ with message length k and codeword length n is a k-dimensional subspace of $\mathbb{F}_2^n$. The messages specify each element of the subspace and the codewords are their representations in $\mathbb{F}_2^n$. Given two codewords $\mathbf{v} = (v_1, v_2, \ldots, v_n)$, and $\mathbf{w} = (w_1, w_2, \ldots, w_n)$, with $v_i, w_i \in \mathbb{F}_2$, the Hamming distance between the two words, denoted by d_H, is the number of coordinates in which $\mathbf{v}$ and $\mathbf{w}$ differ. The minimum distance d_{min} of a linear code $\mathcal{C}$ is the smallest Hamming distance between any two different codewords in $\mathcal{C}$. For linear codes the minimum distance is equal to the minimum non-zero weight in $\mathcal{C}$. We write an $[n, k, d]$-code to mean a binary code $\mathcal{C}$ of length n, cardinality 2^k (encoding messages of length k), and minimum distance d. A linear code with minimum distance d has error correcting capability or error correcting distance $t = \left\lfloor \frac{d_{\text{min}}-1}{2} \right\rfloor$. We assume also the existence of efficient encoding and decoding algorithms for the specific block code chosen. We refer the reader to [2, 22] as standard references for error correcting codes.

Example 1. One of the simplest linear binary error correcting codes is the odd repetition code, $[n, 1, n]$-code. For example, the $[5, 1, 5]$-code would encode the bit $'1'$ into the codeword $(1, 1, 1, 1, 1)$ and the bit $'0'$ into the codeword $(0, 0, 0, 0, 0)$. Decoding is done by simple majority logic. In other words, count the number of one ($'1'$) bits and if there are more than half the number of bits in the whole word, decode to a one and otherwise decode to zero. Thus, the need to encode a bit into an *odd* number of bits.

2.2 Randomness Extraction and Universal Hash Functions

In general, physical sources of randomness, such as Physical Unclonable Functions, are not perfect. By not perfect, we mean that they produce strings of bits which could be biased and which, often, are not uniformly distributed. Notice that cryptographic keys have a general requirement to be both random and uniformly distributed. Thus, a randomness extractor can be used to transform imperfect sources of randomness into random and uniformly distributed strings suitable for cryptographic applications. A possible construction for a randomness extractor is via the concept of universal hash functions.

A universal hash function, introduced by Carter and Wegman in [6], is a map from a finite set A of size a to a finite set B of size b. For a given hash function h and two strings x, x' with $x \neq x'$, we define the function $\delta_h(x, x')$ as equal to 1 if $h(x) = h(x')$ and 0 otherwise. For a finite set (or family) of hash functions $\mathcal{H}$, $\delta_{\mathcal{H}}(x, x')$ is defined to be $\sum_{h \in \mathcal{H}} \delta_h(x, x')$. In other words, $\delta_{\mathcal{H}}(x, x')$ counts the number of functions $h \in \mathcal{H}$ for which x and x' collide. For a random $h \in \mathcal{H}$ and any two distinct x, x', the probability that $h(x) = h(x')$ is $\delta_{\mathcal{H}}(x, x')/|\mathcal{H}|$, where $|\mathcal{H}|$ denotes the size of the set $\mathcal{H}$. There has been extensive research on universal hash functions (see for example [31, 25]). To our knowledge, the work of [19] and the recent work of Kaps et al. [17] are the only ones that consider their hardware implementation.

Example 2. In [6], one of the universal hash function construction suggested is as follows. Take two sets $A = \{0, 1, \cdots, a-1\}$ and $B = \{0, 1, \cdots, b-1\}$, let p be a prime such that $a \leq p$. Define two functions $f_{m,n}(x) = mx + n \bmod p$ for some prime p, where $x \in A$ and $g(y) = y \bmod 2^k$, where $b = 2^k$. Then, the composition $h_{m,n}(x) = g(h_{m,n}(x))$ is a universal hash function. In plain words, you take two random elements of $\mathcal{Z}_p$ (call them m and n) compute $y = f_{m,n}(x)$ take the k least significant bits of y and that is the hash of the string x.

3 PUFs and Helper Data Schemes

In 2001, Pappu et al. [28] introduced the concept of Physical Random Functions or Physical Unclonable Functions (PUFs). Physical Unclonable Functions consist of inherently unclonable physical systems. When a stimulus is applied to the system, it reacts with a response. Such a pair of a stimulus C_i and a response R_i is called a *challenge-response* pair (CRP). In particular, a PUF is considered as a function that maps challenges to responses. Thus, we write: $R_i \leftarrow \mathrm{PUF}(C_i)$. PUFs have essentially two parts: i) a physical part and ii) an operational part. The physical part is a physical system that is very difficult to clone. It inherits its unclonability from uncontrollable process variations during manufacturing. In the case of PUFs on an IC such process variations are typically deep-submicron variations such as doping variations in transistors. The operational part corresponds to a circuit design to take care of the noise present in PUF responses as well as their non-uniform nature. Examples of PUFs include optical PUFs [28], silicon PUFs [10] and coating PUFs [32]. In [11] the notion of an *Intrinsic* PUF or IPUF (a PUF inherently present in a device) was introduced targeting FPGAs. In [14], a similar idea is presented on an ultra-low power chip used in sensor node applications. Recently, [35] has introduced ultra-low cost identifiers based on randomized LC-circuits.

3.1 PUF Security Properties.

As in any security system, in order to evaluate the security of the system, it is necessary that we state the necessary assumptions for the system to be secure. Previous works [28, 10, 32, 12, 11] have either explicitly or implicitly made the following assumptions: (i) It is assumed that a response R_i (to a challenge C_i) gives only a small amount of information on another response R_j (to a different challenge C_j) with $i \neq j$ and (ii) Without having the corresponding PUF (i.e. the actual physical device or structure) at hand, it is impossible to come up with the response R_i corresponding to a challenge C_i, except with negligible probability. In most cases, it is also reasonable to assume that PUFs are tamper evident. This implies that when an attacker tries to investigate the PUF to obtain detailed information about its structure, the PUF is damaged and the challenge-response behavior is changed substantially. It is often assumed as well [12, 11] that the PUF response is only available inside the device after the enrollment procedure. We will also assume this implicitly.

3.2 Helper Data Algorithms

As a result of the noisy nature of PUF responses a Fuzzy Extractor or Helper Data algorithm is required to extract secure keys from them. For formal definitions of Fuzzy Extractors and Helper Data algorithms we refer to [8, 21]. Informally, we need to implement two basic primitives: (i) *Information Reconciliation* or error correction and (ii) *Privacy Amplification* or randomness extraction. In order to implement those two primitives, helper data W are generated during the *enrollment phase* and procedures Gen and Rep are run. In order to implement the procedures Gen and Rep an error correction code $\mathcal{C}$ and a set $\mathcal{H}$ of universal hash functions [6] is required. The Gen-procedure takes as input a PUF response(s) R and produces as output a key K and helper data $W = (W_1, W_2)$. This is achieved as follows. First, a code word $C_S \leftarrow \mathcal{C}$ is chosen at random from $\mathcal{C}$. Then, a first helper data vector equal to $W_1 = C_S \oplus R$ is generated. Furthermore, a hash function h_i is chosen at random from $\mathcal{H}$ and the key K is defined as $K \leftarrow h_i(R)$. The helper data W_2 is set to i. During the key reconstruction phase the procedure Rep is run. It takes as input a noisy response R' from the same PUF and helper data W and reconstructs the key K *i.e.* $K \leftarrow \mathsf{Rep}(R', W)$. This is accomplished according to the following steps: (1) *Information Reconciliation*: Using the helper data W_1, $W_1 \oplus R'$ is computed. Then, the decoding algorithm of $\mathcal{C}$ is used to obtain C_S. From C_S, R is reconstructed as $R = W_1 \oplus C_S$; and (2) *Privacy amplification*: The helper data W_2 is used to choose the correct hash function $h_i \in \mathcal{H}$ and to reconstruct the key as $K = h_i(R)$. Notice that we have implicitly assumed the use of a binary code. This construction is a variant of [16] where the focus was on biometric applications. The security of such constructions has been established in [16, 21, 8, 5].

4 Security for In-Vehicle Systems

In this section we provide an overview of current solutions, attacker models and typical threats for in-vehicle systems. We based our discussion heavily on [38], which provides a recent state-of-the-art survey of these issues.

4.1 Assumptions, Attackers, and Constraints

We will divide attackers in the automotive domain into four categories according to their targets as:

1. *Attackers aiming to modify the in-vehicle system infrastructure.* These attackers can include the vehicle's owner who is interested in modifying the tachometer in order to sell the vehicle for a higher value or higher tax return, manipulating the motor control unit for unauthorized driving parameters, etc. This can also include a corrupt mechanic who, for example, would like to earn extra money by replacing original parts by lower quality ones during an inspection.
2. *Attackers aiming to steal intellectual property (IP).* To these group belong mostly organized crime organizations aiming to produce marketable counterfeits at reduced prices. Counterfeits can include both software and hardware components. This group can also include attackers aiming to steal competitor's expertise by reverse engineering installed vehicle parts, for example.
3. *Attackers whose aim is the vehicle's theft.* These correspond to the well known car robber, who simply obtains access to a vehicle in an unauthorized manner and drives away with it.
4. *Attackers aiming to misuse the vehicle-to-road-to-vehicle infrastructure.* There are several initiatives in Europe [7, 33] that are looking at the types of applications enabled by car-to-car communications and intelligent infrastructure. Clearly, an infrastructure relaying safety, location, and other security critical information will potentially be the target of attacks for personal gain.

Based on their threat analysis against vehicle systems, Wolf et al. [38] have identified the following overall security objectives:

1. **Data confidentiality:** unauthorized access to data considered confidential should not be feasible.
2. **Data integrity:** unauthorized data modification should be infeasible and when feasible, it should be detectable.
3. **Hardware and software component integrity:** unauthorized modifications to vehicular hardware and software components should be infeasible or at least detectable by the vehicle.
4. **Service and data availability:** authorized hardware and software components should be granted access to data and services.
5. **Uniqueness:** hardware components should be infeasible to clone. If a cloning attempt is performed it should be detected and appropriate measures taken.

A defining characteristic of automotive systems is that attackers usually have full physical access to the system and thus have much more freedom to attack the system. Notice that this is in sharp contrast to PC-based IT system where the attacker has no physical access to the system being attacked [38]. In addition to the added adversarial abilities that automotive systems must be able to withstand, their resources are rather limited in terms of processing power, memory, etc. Finally, vehicular systems must be able to work under extreme environmental conditions and for long periods of time (over 20 years), they have very limited connectivity to the outside world (for example, it seems unlikely that you will be able to connect to an online server to receive frequent software or cryptographic material updates), the functionality of the vehicle should work properly even if external communications are severely limited and infrequent, and from a user interaction point of view, almost all vehicular applications are required to run almost completely autonomously. We refer to [38] for an extended discussion of these constraints as well as non-technical ones.

We end this section by noticing that most solutions to the security problem in automotive systems assume the existence of a security module (SM) (also called in [38] security anchor) which provides security relevant functionality such as encryption, decryption, signature generation and verification, hash computation, and secret-key storage. Both hardware and software based solutions are possible as noticed in [38]. As usual hardware based solutions tend to be more secure at higher cost and (somewhat) reduced flexibility when compared to software based ones. It is important to point out that such SM must satisfy certain security requirements. In particular, it should be unclonable, it must be able to be used for secure key storage applications, it must be able to perform cryptographic operations in an efficient manner and without leaking secret information, and it must be able to raise an alarm in case of a security breach [38]. We observe that PUFs thus seem well suited for use inside the SM thanks to their unclonability and tamper evidence properties. We will elaborate on how to use PUFs for automotive applications in Sect. 5. In the following, we review existing published security solutions for automotive applications.

4.2 Existing Schemes

In recent years, numerous articles have appeared focusing on embedded security in vehicles [20, 36, 1, 37, 4, 30]. Work has focused on the areas of component identification, vehicular software protection, and on secure in-vehicle communications. We observe that the types of attacks performed on vehicle systems have also changed over time. In [30], Scheibel et al. propose an architecture for vehicular software protection, in which software is bound to a certain vehicle hardware and software configurations. This guarantees that a content provider's content is only accessible to previously authorized vehicles with the appropriate secret keys (used for decryption of the content) The proposed architecture is based on virtualization technology, the Turaya security kernel, trusted computing (TC) functionality, and on an interoperable legacy operating system. The Turaya security kernel is a small software layer that provides an abstract interface to the hardware resources, enforces strong isolation of applications and implements elementary security services built on a hardware layer, including the support of TC technology. The core of the TC technology is a Trusted Platform Module (TPM), which is considered tamper proof and bound to the particular computing platform.

Adelsbach et al. [1], proposed a protocol for the secure distribution and installation of the software (SW) in an embedded system using public-key broadcast encryption and trustworthy computing to bind the software to the specific embedded system. Wolf et al.[37] emphasized the importance of secure communication inside the vehicle as an enabler of future services in the automotive industry. By considering feasible malicious attacks, they proposed several solutions to various vehicular bus security problems such as authentication, secrecy, etc., based on current modern cryptographic techniques. Bogdanov et al.[4] present a survey of various architectural security solutions for the automotive applications based upon hardware modules such as a customized security controller, TPMs, security boxes, FPGAs and ASICs. They present three possible architectures for the vehicle security system: centralized, semi-centralized, and fully distributed. Each architecture has advantages and disadvantages ranging from increased security in the distributed architecture to less complexity in the centralized one.

In [36], a vehicle component identification scheme is proposed in order to prevent the illegal manipulation, counterfeiting, and exchange of devices. Such component identification can be used then as a basis for future innovative technologies such as electronic license plates, provided that each vehicle has a unique digital identifier. The proposed protocol is based on a central hardware security module (HSM), which is considered to be a tamper proof microcontroller or a TPM module. To identify a component, a passive RFID transponder (with the capability of performing symmetric cryptography) is used. This RFID tag is attached to each security or safety critical component in such a way that removing the tag destroys the component. This can be seen equivalent to the device's ID being unclonable. Each component holds certificate consisting of (PK_C, ID_C) and it has a private key SK_C, where PK_C is the public key corresponding to the private key SK_C, while ID_C is the component's unique identifier. The HSM, imprinted with a secret key K_V, holds a list of all vehicle components, referred to as UL. UL is considered to be securely synchronized regularly with a global list-GL, of all components. The HSM checks the new component certificate on the installment. On provision of the correct certificate, and being not on the GL for another vehicle, secret key K_V is shared with the component. In the proposed protocol it is assumed that the components knowing the secret key K_V are trustworthy and play fair, i.e. they don't compromise the system. The component identification scheme defines three stages during the life-cycle of a component:

(i) *Initialization.* This includes the initial installation of the HSM as well as installation of other components. The HSM can host several types of keys: (1) a randomly chosen key at installation time, which will become the vehicle key K_V. The vehicle key K_V is assumed to be distributed to all components in a secure environment; (2) unique keys shared with critical vehicle components, which can be agreed upon using a Diffie-Hellman key exchange, used to either disable the car, raise an alarm, or display warning messages in case of a system break. During the installation process, it is checked whether the component knows the vehicle key K_V and if so a challenge-response protocol is performed to verify knowledge of the correct key. If the device is new (it does not know K_V), the validity of the component's certificate is checked and if this check passes the vehicle key K_V is sent to the component encrypted with the public key of the component.

(ii) *Running System.* This executes a system check after a pre-determined period of time. The aim is to check that the components have not been manipulated, demounted, replaced, etc. Two methodologies are suggested to perform the check: the HSM challenges each component to prove knowledge of K_V and each component checks another component starting with the HSM until the HSM is challenged by another component, thus closing the authentication loop. As noticed in [36], system checks are vulnerable to compromised components. Moreover, in the authentication loop version, a system component might short-cut the loop skipping a compromised component, for example.

(iii) *Component Demounting.* Two cases can be distinguished: (1) a component is removed and re-installed in the same car, in which case no additional protocol needs to be performed; and (2) a component is removed in an unauthorized manner. To guarantee that (2) does not happen, the authors in [36] present a scheme where the component identification information is removed from the lists UL and GL.

5 Applications of PUFs and Helper Data Algorithms in Vehicular Security

5.1 Secure Key Storage and Component Identification

A key observation in [32] is that the coating can be used to store keys (rather than as a challenge-response repository as in previous works) and that these keys are not stored in memory. Rather, whenever an application requires the key, the key is generated on the fly. This makes it much more difficult for an attacker to compromise key material in security applications. Finally, Tuyls et al. [32] show that active attacks on the coating can be easily detected, thus, making it a good countermeasure against probing attacks. Observe that the use of PUFs as key storage mechanisms is not limited to coating PUFs. In particular, a key property of the helper data algorithm is that given only the helper data or the PUF response, the key can not be derived. In addition, no information can be derived from the public helper data. Thus, making either the PUF response or the helper data inaccessible to an attacker allows for secure key storage. This implies that intrinsic-PUFs in the sense of [11] could be highly suitable as SRAM memory is present in many (if not all) embedded processors. An additional advantage of such SRAM-based PUF is that there are not additional manufacturing steps or modifications necessary as SRAM memory is a standard building block of most embedded systems. Summarizing, secret-key storage with PUFs has several advantages including:

– The secret-key is not available in memory except for a small period of time (when used as input to a encryption/decryption/signature algorithm). This also means that if an attacker opens up the chip where the PUF is contained, it is unlikely that he will be able to obtain any information leading to the recovery of the key.
– Because of the tamper-evidence property of many PUFs, opening a chip to actively and invasively attack it, will most likely result in changes to the physical structure defining the PUF. As a result, an invasive attack will likely result in the key value changing thus, making it impossible to perform any cryptographic operation involving the original key.
– Traditionally, the previous two properties are associated with expensive hardware components. In contrast, PUFs offer this functionality at low cost since it involves semiconductor components which are standard in the building of embedded processors, ICs, and FPGAs.

Observe that a PUF can also be used as an unclonable identifier. In what follows, we show that using a PUF in the system proposed in [36] can result in simplifications of the complexity of the overall system. Weimerskirch et al. [36] proposed to attached an RFID tag to each security or safety critical component in a vehicle for purposes of identification. In addition and as mentioned in Sect. 4.2, each component has both a public and associated private key pair. Notice that the scheme of [36] is based on proving knowledge of secret keys (resp. private keys) using Message Authentication Codes (MACs) (resp. signature schemes). These are well known techniques [23, Chapters 10-11]. PUFs can be used in two complementary manners. First, a PUF can be used to derive the identifier ID, which if implemented via a PUF can be made unclonable as well. This has the advantage that the identifier information is present *intrinsically* in the component and no additional RFID tag needs to be added. Second, the private-public key pair can be derived from a PUF as follows. We assume that the system parameters are available in non-volatile memory of the component in question. For purposes of illustration we consider a system based on elliptic curves [24, 18] (see [3, 13] for thorough treatments of the subject). By system parameters, we mean for example, the coefficients of an elliptic curve, a point P of large order and the order of the subgroup generated by P. Then, a component containing a PUF can be such that it derives a string inside the component and sets this string equal to the private key SK_C of the component, the device stores the corresponding helper data in non-volatile memory, computes the corresponding public key $PK_C = SK_C \cdot P$, and publishes the public key in a certificate also stored in non-volatile memory. Such a system has the advantage that the private key of the component is never known outside the device. In particular, only the device itself knows the private key SK_C. Notice that the advantages of the combination of PUFs with public-key schemes have also been explored in the context of IP protection schemes for FPGA-based systems [12].

5.2 IP Protection

Looking at the uses of software in current and future vehicle systems, we can observe two types of IP:

1. IP aimed at determining the functionality of the vehicle system. For example, automobile manufacturers deploy the same hardware (motor) but define the level of performance for a specific model in software, charging more for automobiles with additional horse power. It is clear that an attacker would be interested in buying a cheaper car model and upgrading the software to obtain the performance of a more expensive model.
2. Third party IP. For example, vehicle navigation systems, and more generally what is termed infotainment in [38]. Services desired on the IP provider side include: time-limited utilization, quantity-limited utilization, device-bound utilization, usage metered utilization, subscription services.

A key property desired in both cases is the ability to limit the number of platforms on which the software (regardless of its use) is running. In the next we show how PUFs can be used to this end. The protocol shown here is based on the protocols used for IP protection on FPGAs presented in [11]. We limit our exposition to the symmetric-key case but public-key based protocols are equally possible [12]. We observe that the only changes in the protocol of [11] are in the parties involved. In particular, in our scenario, we talk about the end-user (USR) as opposed to an IP integrator. IP-provider (IPP), the component's hardware manufacturer (HWM) and the trusted third party (TTP) are available in both cases. We have also included the actual device (DEV) as a party involved in an exchanged with the user

1. **Assumptions:**
   - Communication channel between USR-TTP and TTP-IPP are authenticated and secure
   - Fully trusted TTP

2. **Enrollment Protocol:**
   (a) HWM $\rightarrow$ TTP : $ID_{HWi} || \{\{C_i^1, R_i^1\}\}, \dots \{C_i^n, R_i^n\}\}$
   (b) TTP : Generates helper data W_i^j, W_i^k and corresponding keys K_{ij}, K_{ik} from challenge-response pairs $\{C_i^j, R_i^j\}$ and $\{C_i^k, R_i^k\}$, respectively.

3. **Authentication Protocol (Online):**
   (a) USR $\rightarrow$ TTP : The end-user sends sends request for software ID_{SW} to run on hardware platform ID_{HWi} to the TTP.
   (b) TTP $\rightarrow$ IPP : The TTP forwards the request to the IPP with the software identification number ID_{SW}.
   (c) TTP $\leftarrow$ IPP : The IPP sends the SW to the TTP.
   (d) TTP : The TTP encrypts the SW as $D \leftarrow \mathtt{Enc}_{K_{ij}}(SW || ID_{SW})$ and computes a MAC over the encryption with the second key K_{ik} as $F \leftarrow \mathtt{MAC}_{K_{ik}}(W_i^j || W_i^k || C_i^j || C_i^k || D)$.
   (e) USR $\leftarrow$ TTP : $W_i^j || W_i^k || C_i^j || C_i^k || D || F$.

4. **Authentication Protocol (Offline):**
   (a) DEV$_{HW_i}$ $\leftarrow$ USR : Performed repeatedly whenever the device wants access to the content $W_i^j || W_i^k || C_i^j || C_i^k || D || F$.

Fig. 1. Adapted authentication protocol of [11] with fully trusted TTP

when the system is offline. Figure 1 shows the protocol adapted to the automobile setting. The basic idea in Figure 1 is to use the PUF as a source for secret-key material, both for encryption and MAC-based authentication. During an enrollment procedure the HWM sends a set of challenge-response pairs to the TTP, who then can generate, helper data W_i^j and a corresponding secret key K_i^j. As explained in [11], the MAC (Message Authentication Code) is necessary to authenticate the origin of the IP, since encryption does not provide sufficient authentication guarantees. We emphasized with the *offline* authentication step that the last step is performed without access to an online TTP.

5.3 PUFs as Seals for Insurance Applications

Aftermarket body parts sold to the collision repair industry is a lucrative market for both car manufacturers and independent part manufacturers. In [9], Frost & Sullivan reports that the number of reported collisions has been on the rise, ranging between 9 million and 13 million annually. They also estimate that the number of unreported collisions is climbing every year and it could represent an additional seven million to eight million annually. Independently manufactured aftermarket parts tend to be of lower cost in the range 15% to 45% below original equipment (OE) component prices. This is often due to the lower quality of parts and processes used to manufacture the components. This is especially a lucrative market since some consumers would rather not report to the insurance agency about an accident due to increased premiums and instead get it repaired themselves with cheap parts. This leads to big safety concerns for all road users and additional costs when such cars are involved in accidents. We propose a new system in which insurance agencies can hold consumers responsible for not reporting accidents and give more incentives to the ones which use reliable parts when involved in an accident. This involves building upon an existing system which is used to certify parts and to add a PUF based solution to prove the use of such parts after an accident.

One such system to certify parts is the Certified Automotive Parts Association (CAPA), a non-profit organization created in 1987. After rigorous testing, CAPA certifies that a particular part meets CAPA's recognized quality standards. The manufacturer is then authorized to place CAPA's certification mark on the external body of the certified part. CAPA claims thats once its seal is affixed to a surface, it will self-destruct when removed. This ensures that a seal cannot be transferred from a certified part to a non-certified part. This is an open system which includes both independent and OE manufactures. CAPA has also set up tracing system using a unique serial number on its seals which can be used by anyone (not just repair shop or insurance companies) online. However, it is not a cradle-to-grave tracing as the CAPA server do not store any information on when and where the parts have been used. Misuse such as illicit placing of such a seal on a non-certified part is prevented only by legal means. This creates a big loop-hole for counterfeit manufacturers who can easily access the public database to create seals with appropriate numbering. The lack of cradle-to-grave tracing also prevents the insurance agencies to accurately judge if a part was used on car before or after an accident.

Our solution involves first adding a PUF based seal to make the identifier unclonable by non-certified parties and counterfeiters. The vehicle system is similar to the one suggested in [36] in which a central HSM identifies and holds the status of all components attached to the system. The HSM's main goal is to store a status image of the components attached to the car at different time intervals. After an accident, the insurance agency can irrefutably determine the components that were attached during an accident based on the certified copy of the tamper resistant HSM.

6 Conclusions

The present contribution shows the use of PUF to solve various vehicle system security problems. We first gave an overview of the various in-vehicle system security issues and existing solutions. We showed how PUF based secret-key storage can be used to enhance and simplify existing solutions. The property of inherent unclonability gives advantages for anti-counterfeiting. On the fly secret key generation using PUFs provides added security compared to existing solutions. We also show that how to bind software to a device containing a PUF identifier, thus resulting in a secure IP protection system. Finally, We also discuss a novel application of PUFs as a seal for insurance agencies to determine the quality of parts used on a vehicle during an accident in an irrefutable way.

References

1. A. Adelsbach, U. Huber, and A.R. Sadeghi. Secure Software Delivery and Installation in Embedded Systems. In *Information Security Practice and Experience*, volume 3439 of *LNCS*, pages 255–267, Seattle, USA, July 2006. Springer.
2. R. E. Blahut. *Theory and Practice of Error Control Codes.* Addison-Wesley Publishing Company, first edition, 1985.
3. I. Blake, G. Seroussi, and N. P. Smart. *Elliptic Curves in Cryptography.* London Mathematical Society Lecture Note Series. Cambridge University Press, 1999.
4. A. Bogdanov, D. Carluccio, A. Weimerskirch, and T. Wollinger. Embedded Security Solutions for Automotive Applications. In *Advanced Microsystems for Automotive Applications*, pages 177–191. Springer-Verlag, 2007.
5. X. Boyen, Y. Dodis, J. Katz, R. Ostrovsky, and A. Smith. Secure remote authentication using biometric data. In R. Cramer, editor, *Advances in Cryptology — Eurocrypt 2005*, volume 3494 of *LNCS*, pages 147–163. Springer-Verlag, 2005.
6. L. Carter and M. N. Wegman. Universal Classes of Hash Functions. *J. Comput. Syst. Sci.*, 18(2):143–154, 1979.
7. CAR 2 CAR Communication Consortium. Available at `http://www.car-to-car.org/`. Accessed on July 1st, 2008.
8. Y. Dodis, M. Reyzin, and A. Smith. Fuzzy extractors: How to generate strong keys from biometrics and other noisy data. In C. Cachin and J. Camenisch, editors, *Advances in Cryptology —- EUROCRYPT 2004*, volume 3027 of *LNCS*, pages 523–540. Springer-Verlag, 2004.
9. North American Collision Replacement Body Panel Aftermarket. Research service report, Frost & Sullivan, 14 Feb 2005. Available at `http://www.frost.com/prod/servlet/report-brochure.pag?id=F220-01-00-00-00`.
10. B. Gassend, D. E. Clarke, M. van Dijk, and S. Devadas. Silicon physical unknown functions. In V. Atluri, editor, *ACM Conference on Computer and Communications Security — CCS 2002*, pages 148–160. ACM, November 2002.
11. J. Guajardo, S. S. Kumar, G.-J. Schrijen, and P. Tuyls. FPGA Intrinsic PUFs and Their Use for IP Protection. In P. Paillier and I. Verbauwhede, editors, *Cryptographic Hardware and Embedded Systems — CHES 2007*, volume 4727 of *LNCS*, pages 63–80. Springer, September 10-13, 2007.
12. J. Guajardo, S. S. Kumar, G.-J. Schrijen, and P. Tuyls. Physical Unclonable Functions and Public Key Crypto for FPGA IP Protection. In *International Conference on Field Programmable Logic and Applications — FPL 2007*, pages 189–195. IEEE, August 27-30, 2007.
13. D. Hankerson, A. Menezes, and S. Vanstone. *Guide to Elliptic Curve Cryptography.* Springer, 2004.
14. D. E. Holcomb, W. P. Burleson, and K. Fu. Initial SRAM state as a fingerprint and source of true random numbers for RFID tags. Conference on RFID Security 07, July 11-13, 2007.
15. D. M. Hopkins, L. T. Kontnik, and M. T. Turnage. *Counterfeiting Exposed: Protecting your Brand and Customers.* Business Strategy. Wiley, 2003.

16. A. Juels and M. Wattenberg. A Fuzzy Commitment Scheme. In J. Motiwalla and G. Tsudik, editors, *ACM Conference on Computer and Communications Security — ACM CCS '99*, pages 28–36. ACM, November 1-4, 1999.
17. J.-P. Kaps, K. Y., and B. Sunar. Energy Scalable Universal Hashing. *IEEE Trans. Computers*, 54(12):1484–1495, 2005.
18. N. Koblitz. A family of jacobians suitable for discrete log cryptosystems. In S. Goldwasser, editor, *Advances in Cryptology - CRYPTO '88*, volume 403 of *LNCS*, pages 94–99. Springer, 1988.
19. H. Krawczyk. LFSR-based Hashing and Authentication. In Y. Desmedt, editor, *Advances in Cryptology - CRYPTO '94*, volume 839 of *LNCS*, pages 129–139. Springer, August 21-25, 1994.
20. K. Lemke, A.R. Sadeghi, and C. Stüble. An open approach for designing secure electronic immobilizers. In *Proceedings of the 1st International Conference on Information Security Practice and Experience ISPEC '05*, pages 230–242, Singapore, April 2005.
21. J.-P. M. G. Linnartz and P. Tuyls. New Shielding Functions to Enhance Privacy and Prevent Misuse of Biometric Templates. In J. Kittler and M. S. Nixon, editors, *Audio-and Video-Based Biometrie Person Authentication — AVBPA 2003*, volume 2688 of *LNCS*, pages 393–402. Springer, June 9-11, 2003.
22. F. J. MacWilliams and N. J. A. Sloane. *The Theory of Error-Correcting Codes.*, volume 16 of *North-Holland Mathematical Library.* North-Holland/Elsevier, Amsterdam, The Netherlands, 1977.
23. A. Menezes, P. van Oorschot, and S. Vanstone. *Handbook of Applied Cryptography.* CRC Press, 1997.
24. V. S. Miller. Use of elliptic curves in cryptography. In H. C. Williams, editor, *Advances in Cryptology - CRYPTO '85*, volume 218 of *LNCS*, pages 417–426. Springer, 1985.
25. W. Nevelsteen and B. Preneel. Software Performance of Universal Hash Functions. In J. Stern, editor, *Advances in Cryptology — EUROCRYPT'99*, volume 1592 of *LNCS*, pages 24–41. Springer, May 2-6, 1999.
26. E. Nickel. IBM automotive software foundry. in Press Conference on Computer Science in Automotive Industry, September, 2003.
27. E. Nickel. ADT/tyco fire and security, a. technology, i. inc., i. corporation, s. t. inc., and xterprice. RFID and UHF: A prescription for RFID success in the pharmaceutical industry. White paper, Pharmaceutical Online. Available at `http://www.Pharmaceuticalonline.com/uhf/`, June 2006.
28. R. S. Pappu. *Physical one-way functions.* PhD thesis, Massachusetts Institute of Technology, March 2001. Available at `http://pubs.media.mit.edu/pubs/papers/01.03.pappuphd.powf.pdf`.
29. A. Saad and U. Weinmann. Automotive software engineering and concepts. In K. R. Dittrich, W. König, A. Oberweis, K. Rannenberg, and W. Wahlster, editors, *INFORMATIK 2003 - Innovative Informatikanwendungen, Band 1, Beiträge der 33. Jahrestagung der Gesellschaft für Informatik e.V. (GI Jahrestagung (1))*, volume 34 of *LNI*, pages 318–319. GI, 29. September - 2. Oktober, 2003.
30. M. Scheibel, C. Stüble, and M. Wolf. An Interoperable Security Architecture for Vehicular Software Protection. In F. Michahelles, editor, *1st International Workshop on Interoperable Vehicles, IOV*, September-October 2008.
31. V. Shoup. On Fast and Provably Secure Message Authentication Based on Universal Hashing. In N. Koblitz, editor, *Advances in Cryptology - CRYPTO '96*, volume 1109 of *LNCS*, pages 313–328. Springer, August 18-22, 1996.
32. P. Tuyls, G.-J. Schrijen, B. Škorić, J. van Geloven, N. Verhaegh, and R. Wolters. Read-Proof Hardware from Protective Coatings. In L. Goubin and M. Matsui, editors, *Cryptographic Hardware and Embedded Systems — CHES 2006*, volume 4249 of *LNCS*, pages 369–383. Springer, October 10-13, 2006.
33. UbiSec&Sens — Ubiquitous Sensing and Security in the European Homeland. Available at `http://www.ist-ubisecsens.org/`. Accessed on July 1st, 2008.
34. B. Škorić, P. Tuyls, and W. Ophey. Robust Key Extraction from Physical Uncloneable Functions. In J. Ioannidis, A. D. Keromytis, and M. Yung, editors, *Applied Cryptography and Network Security — ACNS 2005*, volume 3531 of *LNCS*, pages 407–422, June 7-10, 2005.
35. B. Škorić, T. Bel, A.H.M. Blom, B.R. de Jong, H. Kretschman, and A.J.M. Nellissen. Randomized resonators as uniquely identifiable anti-counterfeiting tags. Technical report, Philips Research Laboratories, January 28th, 2008.
36. A. Weimerskirch, C. Paar, and M. Wolf. Cryptographic component identification: Enabler for secure vehicles. In *IEEE 62nd Vehicular Technology Conference, VTC-2005-Fall*, volume 2, pages 1227–1231, Singapore, September 25-28, 2005. IEEE.
37. M. Wolf, A. Weimerskirch, and C. Paar. Secure In-Vehicle Communication. In K. Lemke, C. Paar, and M. Wolf, editors, *Embedded Security in Cars*, pages 95–109. Springer, 2006.
38. M. Wolf, A. Weimerskirch, and T. Wollinger. State of the art: Embedding security in vehicles. *Eurasip Journal on Embedded Systems*, Vol. (2007), 2007.