AN14223

MIFARE DUOX for EV charging use cases

Rev. 1.0 — 10 July 2025 Application note

Document information

Information	Content
Keywords	MIFARE DUOX, EV charging, electric vehicle charging, EV charging card, NFC, ISO/IEC 14443, ISO/IEC 7816-4, VDE, DKE, VDE-AR-E 2532-100, ECC, ECDSA, asymmetric cryptography, user authentication, user authorization
Abstract	This document describes how MIFARE DUOX can be used for EV charging applications, to authorize and authenticate the end user in a secure and reliable way. The MIFARE DUOX smart card IC can be used as the RFID medium for the EV charging device, in the form of an NFC smart card, NFC keyfob, NFC token, or similar.

1 Introduction

With the quick expansion of the electromobility market and the growing number of electric vehicles on the road, the demand for robust and reliable electric vehicle charging infrastructure and charging services is rising.

In order to charge electric vehicles at charging stations, some means to authorize the charging session and execute the payment is required. Typical mediums in use today are NFC-based smartcards (also often referred to as "RFID cards" within this industry) and mobile devices with dedicated smartphone apps.

Using smartcards or mobile devices, users can be uniquely identified, and all user-related data can be easily associated with executed charging sessions, billing data and other statistics. This helps tremendously to enhance transparency and user experience for the end customer. With a simple NFC-based tap onto the NFC reader of an EV charging terminal, the charging session is authorized without the need to enter a password, pin, or credit card details. It offers electric vehicle drivers an easy and convenient way to access public charging private charging stations.

EV charging cards can come in any form factor, such as a standard smartcard, tags, tokens, keyfobs, but are all based on the same technology, which is NFC (Near Field Communication), also referred to as RFID (Radio Frequency Identification). EV charging cards can be used with residential chargers to power your vehicle at home, but are more frequently used to gain access to public EV charging, such as stations located on the road, on highways, in cities, at shopping centers, or the workplace.

The main benefit of using an EV charging card (NFC-based smart card) is reliability. The EV charging card operates without battery, purely offline, and is resistant to extreme environment conditions such as heat and cold. In contrast to this, an EV charging smartphone app may not be accepted at charging stations, due to technical limitations or non-compliancy. Furthermore, a smartphone app cannot be used if the phone is out of battery.

2 NFC-based EV charging cards and current market situation

In today’s infrastructure for electric vehicle charging, many charging stations support the use of contactless smartcards, so drivers can stop and start the charging process, make payments, and register data about the charging session with a simple NFC-based tap. Using smartcards at charging stations adds convenience to EV ownership and helps promote wider adoption of EVs.

It also makes it easier for charging networks to provide individual drivers and fleet operators with the kinds of data-driven insights that help monitor vehicle safety and manage power consumption for greater sustainability.

There are, however, concerns over security. In many cases of EV charging installations that we can find in the field today, the charge point relies on the contactless smart card’s unique identifier (UID) for authentication, which is a practice that is vulnerable to fraud, including the use of cloned or fake smartcards. The UID of a contactless smart card uniquely identifies the card and links it to an EV driver through the EV provider's backend system. The user´s account is linked to the UID when the smart card is issued and is used to look up the user account for every charging session. Since the UID can be easily read out from the charging card and retrieved in plaintext, relying solely on it for authorizing the user poses a high security risk, as hackers can clone the user credential, by simply copying the UID, and charge at another driver's expense.

Within this chapter, we want to explore the purpose and benefits of EV charging cards, the different types of cards, as well as the implementation of NFC on currently deployed EV charging stations. The interaction between the NFC reader on the charge point and the EV charging card is essential to ensure security and provide protection of the EV driver´s data.

2.1 Benefits and necessity of using NFC-based EV charging cards

EV charging cards are essential for improving the efficiency of the charging operation as well as the accessibility of the infrastructure used for charging electric vehicles.

Various different kinds of EV charging cards do exist, catering to a diverse range of end customer requirements and market needs. Ranging from home charging, to company charging, fleet management and public charging, NFC-based smartcards can cater to various different use cases.

- **Residential and at-home personal cards**
  These EV charging cards are catering to individual end users who want to access their EV charger from their single-family home, multi-family home, apartment building, or multi-tenant building.
- **Public personal cards**
  These EV charging cards are fulfilling EV driver´s requirements who charge their car across various public chargers located in public charging networks. These can be chargers along roads, highways, cities, shopping centers, public buildings, and more.
  We can distinguish further into network-specific EV charging cards and universal EV charging cards. Network-specific EV charging cards work in one EV charging network that´s served by one CPO (Charge Point Operator), whereas universal EV charging cards work in many EV charging networks which are served by many CPOs. The later scenario is also called EV Roaming.
  EV Roaming essentially connects multiple eMSPs (Electro Mobility Service Providers) with multiple CPOs. By connecting eMSPs and CPOs, the EV charging network coverage can be extended, so that users can charge their EV in a variety of chargers from various different networks, which are served and operated by several different CPOs.
  Universal EV charging cards that support EV Roaming are highly convenient, offering the greatest possible flexibility for the EV driver. By gaining access to various different charging networks inside one country and internationally makes personal EV charging cards invaluable for navigating in the EV world.

2.2 UID - What is it?

Each smart card or NFC tag that adheres to the ISO/IEC 14443 standard, needs to have a UID (unique identifier), which is used during the anti-colission phase of the ISO/IEC 14443 card activation sequence. The UID ensures that if multiple cards are placed on an NFC reader at the same time, the reader can distinguish the cards and only activates one card at a time. By providing a “unique” number, this ensures that in case a “collision” will happen during transmission of this number, that can be detected by the reader.

Historically, UIDs were 4 bytes long, which obviously does not provide a large enough number space to have a unique number on every NFC device globally. Therefore, 4-byte UIDs are reused from time to time, which is why they are also referred to as NUID (Non-Unique IDentifier).

The ISO/IEC 14443 also supports 7-byte and 10-byte long UIDs at the moment, where 7-byte UIDs are most common at the moment. The number space of 7 bytes is big enough so that 7-byte UIDs are considered real unique numbers for NFC devices. Any 7-byte or 10-byte UID issued from NXP can be considered a real unique number. Also, it is ensured that other manufacturers' 7-byte UIDs do not collide with NXP-issued UIDs by using the first byte of the UID (referred to as UID0) as a manufacturer code. Therefore, all NXP issued 7-byte UIDs start with the value 04h.

On every NXP product, the UID cannot be changed and can never be influenced by the user. The UID is encoded during NXP´s manufacturing process in a secure manner.

While a smart card or NFC tag UID might look like a serial number at first glance, it is not meant to be one. The main reason for this is that the UID does not provide any means of checking its validity. Other chip manufacturers may misuse the UID scheme and purposefully use, for example, NXP's manufacturer code 04h in their own products, which in turn may lead to duplicates, or even worse, they might build smartcards that allow the UID to be changed to any number, which effectively allows the cloning of a single UID onto several cards.

Also, devices such as the Flipper Zero, Chameleon Ultra, iCopy-X, or Proxmark, allow the emulation of NFC devices with any UID, which poses a significant risk for systems that rely only on the uniqueness of the UID.

2.3 UID-only approach for EV charging and its security weaknesses

In the last few years, concerns over security related to NFC-based EV charging cards have increased.

In many current EV charging installations, the charging point relies on the contactless smart card’s unique identifier (UID) for authentication, which is a practice that is vulnerable to fraud, including the use of cloned or fake smartcards. The UID of a contactless smart card uniquely identifies the card and links it to an EV driver through the EV provider's backend system. The user's account is linked to the UID when the smart card is issued and is used to look up the user account for every charging session.

Since the UID can be easily read out from the charging card and retrieved in plaintext, relying solely on it for authorizing the user poses a high security risk, as hackers can clone the user credential, by simply copying the UID, and charge at another driver's expense.

2.4 Security challenges for NFC implementations within the EV charging infrastructure

With the charging station relying on only the UID of the EV charging card to authorize a charging session, the access to electricity is practically not protected at all.

As we heard that the UID can be easily read out from the charging card and retrieved in plaintext, relying solely on it for authorizing the user and further authorizing the charging session, poses a high security risk.

The UID can be read out easily by attackers, and used to create a copy of the card.

Since no other data is stored on the EV charging card, everybody can retrieve the card´s UID very easily. No other data besides the UID is required to authorize the user at the charging station.

The UID can then further on be used to be copied to an emulator (a device that simulates a charging card ´s behavior) or a clone card, which allows to change the UID by the hacker. (Typically NFC smartcards have assigned the UID during production, and the UID is not changeable. However, for cloned cards, it is possible to overwrite the UID and set it to any desired value.)

By simply finding out the UID and copying it to a clone card, the hacker can create a 1:1 copy of a valid user credential, and so charge at another driver´s expense.

Creating one or many copies of the same UID, and therefore creating cloned credentials, can lead to catastrophic results for the affected EV driver, as well for the involved eMSPs and CPOs. The potential of fraud and monetary loss is extremely high, given the fact that UIDs can be retrieved so easily from EV charging cards.

Increasing the protection of the user authentication and authorization on the NFC channel is crucial to avoid fraud in the EV charging ecosystem.

Several possibilities for how security can be enhanced are illustrated in Section 3.

3 Security enhancements for NFC-based EV charging

To increase security compared to the UID-only approach, various different possibilities do exist, each posing their individual benefits and difficulties.

In this chapter we want to discuss the different options and their feasibility.

Once the EV driver approaches the EV charging point, and taps the EV charging card via NFC to the reader terminal of the charging point, the NFC session between reader and smart card is started.

As a first step, the UID of the EV charging card is read out. Typically, then the system (online via the connected backend service or offline directly on the charge point) validates the UID of the charging card and authorizes the charging session for the EV driver. As we have established, relying on the UID-only, is very insecure and poses huge security risks.

To improve this situation, the following enhancement is proposed: After reading out the UID of the EV charging card, but before starting the charging session, it is possible to execute one additional step of secure cryptographic authentication between the charge point and the smart card.

Several different authentication possibilities do exist, which are compared in this chapter.

- **Symmetric authentication**
  The authentication step uses symmetric cryptography (for example, AES-128), which is a major step up in terms of protection.
  A mutual symmetric authentication ensures that the charging card has not been copied, as the symmetric key which is used for authentication, can never be extracted from the smart card. Therefore, a successful symmetric authentication ensures that the card which is presented by the end user to the charging point is unique and a valid card to authorize and start the charging session.
  Besides guaranteeing a major step up in security, this approach is still subject to its own set of limitations and vulnerabilities within the complex EV charging ecosystem and infrastructures.
  The concept of symmetric cryptography relies on the fact that the same secret, symmetric key needs to be present on the charging point (reader terminal) and the charging card, which poses its specific difficulties.
  - The EV charging card needs to have a symmetric key injected, which is card-specific.
    Injecting a symmetric key into the EV charging card during card personalization is a simple and fast process, and once the key was injected into the card, the key is stored securely on the card, and there is no way to retrieve it from the device anymore.
  - The EV charging station needs to have a symmetric key injected.
    Injecting a symmetric key into the EV charging infrastructure, into each and every reader terminal of all charging points, is a much more complex process, which requires a high secure key sharing and key distribution mechanism, as well as secure and tamper-resistant hardware on the charging point, to store the sensitive key material (for example, secure key store, secure access module, or similar).

  Symmetric key management is relatively complex, especially when scaling to large numbers of parties involved. Keeping the symmetric key secured at all times, and sharing it considering the right security measures, especially with a big number of players involved can increase the risk of key leakage.
  Especially, in EV Roaming scenarios, where multiple eMPSs and multiple CPOs are connected, the same secret symmetric key needs to be shared with every involved company. This is usually not desired and poses huge key distribution challenges.
- **Mutual Asymmetric authentication**
  The authentication step uses asymmetric cryptography (for example, ECC, RSA), which is a major step up in terms of protection.
  A mutual asymmetric authentication ensures that the charging card has not been copied, as the keys which are used for authentication, can never be extracted from the smart card. Therefore, a successful asymmetric authentication ensures that the card which is presented by the end user to the charging point is unique and a valid card to authorize and start the charging session.

Asymmetric cryptography provides greater protection than the UID-only approach and more flexibility compared to symmetric cryptography. With asymmetric cryptography, security is ensured by pairing a public key, which can be distributed openly without compromising security, with a private key that must be kept confidential.
For mutual asymmetric authentication, both the charging point and the charging card need to possess a private / public keypair which conforms to the requirements of asymmetric cryptography. Mutual asymmetric authentication ensures that both the charging card, as well as the charging point authenticate themselves toward each other.
To realize a mutual authentication, the following applies:

- The EV charging card needs to have a private / public keypair injected, which is card-specific.
  Injecting an asymmetric keypair into the EV charging card during card personalization is a simple and fast process, and once the keys were injected into the card, the keys are stored securely on the card. The public key can further on be retrieved from the card, as the public key can be shared freely without any concern.
  The private key is stored securely on the card, and there is no way to retrieve it from the device anymore. It is only used during authentication and encryption processes, when needed, by the chip itself, but not transmitted anymore via NFC.
- The EV charging station needs to have a private / public keypair injected, which is station-specific.
  Injecting an asymmetric keypair into the EV charging infrastructure, into each and every reader terminal of all charging points, is a partially complex process.
  Injecting the public key and related certificate is straightforward, as they can be shared and circulated publicly, without security concerns. The public key, as well as the certificate, can be stored in normal memory on the charging point, outside of any secure hardware.
  Injecting the private key is a more complex undertaking, as the private key needs to remain secret at all times and requires a high secure key sharing and key distribution mechanism, as well as secure and tamper-resistant hardware on the charging point, to store the sensitive secret key material (for example, secure key store, secure access module, or similar).

- **Unilateral Asymmetric authentication**
  The authentication step uses asymmetric cryptography (for example, ECC, RSA), which is a major step up in terms of protection, as already explained above.
  Also the unilateral asymmetric authentication ensures that the charging card has not been copied, as the keys which are used for unilateral authentication, can never be extracted from the smart card.
  For unilateral asymmetric authentication, only the charging card needs to possess a private / public keypair, whereas the charging point only needs to possess a public key and certificate. Unilateral asymmetric authentication ensures that the charging card authenticates itself toward the charging point. However, the charging point does not authenticate itself toward the card. As we assume that the EV charging infrastructure is trustworthy, and the charging point has not been compromised, this is a great solution for securely authenticating the EV charging card, and proving that it´s authentic.
  To realize a unilateral asymmetric authentication, the following applies:
  - The EV charging card needs to have a private / public keypair injected, which is card-specific.
    Injecting an asymmetric keypair into the EV charging card during card personalization is a simple and fast process, and once the keys were injected into the card, the keys are stored securely on the card. The public key can further on be retrieved from the card, as the public key can be shared freely without any concern.
    The private key is stored securely on the card, and there is no way to retrieve it from the device anymore. It is only used during authentication and encryption processes, when needed, by the chip itself, but not transmitted anymore via NFC.
  - The EV charging station needs to have a public key injected, which is station-specific.
    Injecting a public keypair into the EV charging infrastructure, into each and every reader terminal of all charging points, is not complex and does not pose any security risk.
    Injecting the public key and related certificate is straightforward, as they can be shared and circulated publicly, without security concerns. The public key, as well as the certificate, can be stored in normal memory on the charging point, outside of any secure hardware.
    No injection of a private key is required, so also the complexity related to private key handling is eliminated.

4 EV charging regulation from VDE

Bringing a higher level of security to EV charging applications addresses concerns over fraud, counterfeiting, and data integrity. It also expands the opportunities for multi-application smart card use, with support for functions beyond EV charging, such as micropayments, secure car access, parking access, and more.

Concerns over security have, in recent years, led to various efforts to address security in smartcard-based EV charging. One of the most prominent and widely supported of these efforts is the VDE-AR-E 2532-100 application rule. Issued by VDE, a non-profit service organization concerned with the generation, distribution, and safe use of electricity, and the DKE German Commission for Electrical, Electronic & Information Technologies, the VDE-DKE guidelines aim to prevent unauthorized charging and fraud in the charging ecosystem by upgrading to asymmetric cryptography.

4.1 Purpose of VDE-AR-E 2532-100 application rule

The VDE-AR-E 2532-100 application rule was issued to define requirements and solution proposals for enhancing security during authorization and authentication of an EV charging session at an EV charging point. The purpose of this guideline is to define a concrete implementation proposal for a secure and interoperable authentication procedure toward the EV charging provider via either an RFID device (NFC-based EV charging smart card) or a remote backend service.

The application rule focuses on two main proposals: the upgrade of existing EV charging points and networks in order to introduce more security and a new way of end user authorization, as well as the installation of new EV charging points and new EV charging networks and their installation, utilizing secure technologies and secure implementation.

Special focus was put on being compatible with already existing infrastructure and charging points in the field, in order to ensure compatibility with already working systems. Furthermore, the proposal also covers EV roaming scenarios, ensuring cross-network adaptability.

4.1.1 NFC-based EV charging compliant to VDE-AR-E 2532-100

This section focuses on the utilization of NFC-based EV charging cards or devices which embed a secure contactless smart card chip (also called RFID transponders; RFID mediums) on the technological basis of ISO/IEC 14443-3 and ISO/IEC 14443-4.

The guideline´s focus was put onto adressing the issue of copying a valid EV charging card´s UID and utilizing it to clone a credential or emulate a new card with the same UID. This weakness is addressed via the concept of asymmetric cryptography and the following two means, which can be executed in an offline manner directly on the EV charging station (without any mandatorily required backend connectivity):

- Secure verification of the origin of the EV charging smart card chip
- Secure verification of the uniqueness and authenticity of the EV charging smart card chip

The concept of VDE-AR-E 2532-100 is based on a chain-of-trust concept, starting already at the manufacturer of the smart card chip (semiconductor provider) which will be embedded into the EV charging smart card.

The manufacturer of the smart card chip can be any semiconductor company which can fulfill the requirements of secure chip personalization, secure chip manufacturing, and smart card chip requirements as outlined in Section 4.1.2.1.

4.1.2 Chain of trust and requirements for involved entities

The concept of VDE-AR-E 2532-100 is based on a chain-of-trust concept, starting already at the manufacturer of the smart card chip (semiconductor provider) which will be embedded into the EV charging smart card.

The manufacturer of the smart card chip can be any semiconductor company which can fulfill the requirements of secure chip personalization, secure chip manufacturing, and smart card chip requirements as outlined in Section 4.1.2.1.

4.1.2.1 Requirements for the NFC smart card chip

The smart card chip used for an EV charging card needs to meet following requirements to be compliant with VDE-AR-E 2532-100:

- Certification according to Common Criteria on CC EAL5 or higher
- Support for asymmetric cryptography with ECC (elliptic curve cryptography) including the operations for hashing and signing
- An EV charging application needs to be installed on the chip utilizing the application identifier (AID) 0xA00000084500000000000000000001
- The chip unique asymmetric keypair needs to utilize the ECC 256-bit curve brainpoolP256r1
- Hardware and software of the chip need to provide resistance against common industry attacks like key extraction; chip copying; chip cloning; side-channel attacks; and similar
- Provide mechanism to lock the data and configuration settings which have been applied during personalization
- Meet the requirements of ISO/IEC 14443-4 standard

The VDE-AR-E 2532-100 regulation defines a unilateral card authentication mechanism using an ECDSA-based challenge-response protocol supported by a certificate. The targeted EV charging application on the chip will hold the private key from a key pair on the brainpoolP256r1 curve. Next to this, there are two files: one holding a related certificate, and a second file for additional data. The private key and certificate can be preprovisioned during manufacturing of the semiconductor vendor or semiconductor manufacturer, as required by the standard.

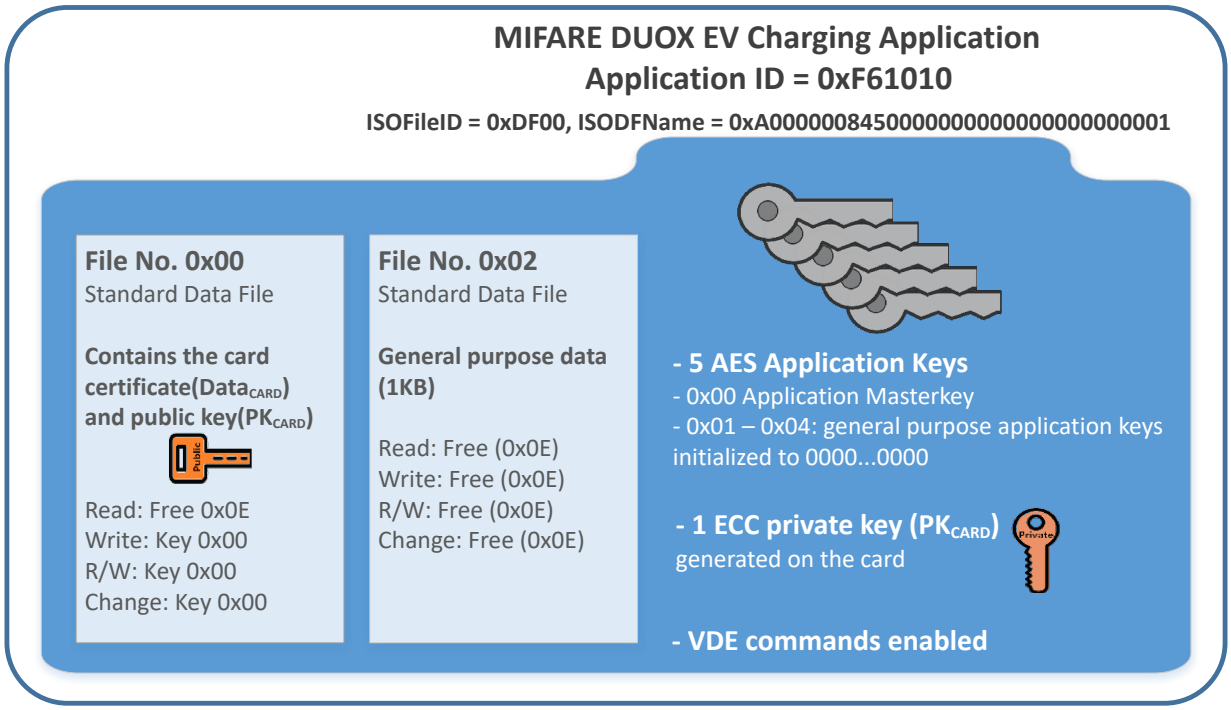


Figure 1. EV charging application on MIFARE DUOX

4.1.2.2 Requirements for the personalization entity

The personalization entity which personalizes the smart card chip used for an EV charging card needs to meet the following requirements to be compliant with VDE-AR-E 2532-100:

- The personalization entity needs to personalize the smart card chip in a secure environment
- The following data needs to be injected into the smart card chip, or generated on the smart card chip, during the personalization:
  - Serial number (unique ID; UID) of the chip (at least 7-byte unique ID)
  - Asymmetric keypair which is unique for the chip: public key (PK$_{CARD}$) and private key (SK$_{CARD}$) - should be directly generated on the chip
  - Digital signature of the personalization entity (Signature$_{VENDOR}$; respectively in the NXP case: Signature$_{NXP}$) signing the chips certificate data (DATA$_{CARD}$). The signature needs to be calculated using the ECDSA-256 algorithm using the ECC-256 curve brainpoolP256r1 and the hashing algorithm SHA256
- The personalization entity is the semiconductor manufacturer. This entity shall own its own asymmetric keypair which is unique: public key (PK$_{VENDOR}$) and private key (SK$_{VENDOR}$)
  - The public key (PK$_{VENDOR}$; respectively in the NXP case: PK$_{NXP}$) shall be made accessible to any interested player from the EV charging ecosystem. This public key shall be injected into every EV charging station which wants to interact with EV charging cards compliant to the VDE-AR-E 2532-100 application rule
  - The private key (PK$_{VENDOR}$; respectively in the NXP case: SK$_{NXP}$) needs to remain secret and kept stored secure in the personalization entity´s backend system

4.1.2.3 Requirements for the EV charging station

The EV charging station which shall interact with the EV charging card needs to meet the following requirements to be compliant with VDE-AR-E 2532-100:

- The EV charging station shall securely import the public key (PK_{VENDOR}; respectively in the NXP case: PK_{NXP}) from the smart card chip vendor / personalizer
- The EV charging station shall implement a software upgrade which is running on the charging station, in order to interact via the NFC interface between the NFC-based reader terminal on the charging station and the NFC-based EV charging card, in order to:
  - Securely verify the origin of the EV charging smart card chip (see details in Section 4.1.2.3.2);
  - And to ensure secure verification of the uniqueness and authenticity of the EV charging smart card chip (see details in Section 4.1.2.3.2).
- In order to ensure the required verification of the charging card, a software update on the charging point is required.
  - Implementation of several ISO/IEC 14443-4 compliant APDUs is required.
  - Cryptographic operations using ECC-256 bit are needed.

4.1.2.3.1 Verification of smart card chip origin

Once an EV driver approaches the EV charging station to start a charging session, its EV charging card will be presented toward the NFC-based reader of the charging station. The following steps shall be executed by the charging station:

1. Detection of the charging card via the NFC-based reader terminal on the EV charging station and activation fo the charging card according to ISO/IEC 14443-4
2. Selection of the EV charging application which is encoded to the charging card using the ISOSelect command with the AID 0xA0000008450000000000000000000001
3. Reading out the certificate data ($DATA_{CARD}$) from the charging card which was signed by the secret key of the personalization entity (SK_{VENDOR}) and also contains the mentioned signature
4. Verification of the certificate data ($DATA_{CARD}$) by using the public key of the personalization entity (PK_{VENDOR}; respectively in the NXP case: PK_{NXP}), which was injected to the charging point already earlier

If steps 1 to 4 were successful, and the signature was validated successfully, the origin and integrity of the smart card chip is guaranteed, and the verification can continue with the next steps as outlined in Section 4.1.2.3.2.

In case any of the above mentioned steps fails, for example, the signature validation is not successful, the authorization process needs to be stopped, and the user will not be authorized to start a charging session.

4.1.2.3.2 Verification of smart card chip uniqueness and authenticity

Once the chip origin was proven via verification of the chip´s unique certificate and signature, as outlined in Section 4.1.2.3.1, following steps shall be executed by the charging station, in order to proof the smart card chip ´s uniqueness and authenticity:

1. Generation of a 32-byte random number at the EV charging station and sending it toward the EV charging card
2. EV charging card receives the 32-byte random number and calculates an SHA-256 has based on the numbers, and signs this has using ECC-256 signature generation using its private key SK_{CARD}
3. EV charging card returns the dynamically generated signature to the EV charging station
4. Verification of signature at the EV charging station using PK_{CARD}

If steps 1 to 4 were successful, and the signature was validated successfully in step 4, the uniqueness of the smart card chip is guaranteed, and the verification of the EV charging card is completed. Now, the EV charging station can authorize the EV charging card for a charging session, by granting charging rights for the UID which was read out in the beginning of the transaction protocol. Allowing the UID to start a charging session can happen either offline or online, depending how the EV charging system is implemented.

In case any of the above mentioned steps fails, for example, the dynamically generated signature validation is not successful, the authorization process needs to be stopped, and the user will not be authorized to start a charging session.

4.1.2.3.3 Implementation of required NFC commands

For realizing the scenarios as outlined in Section 4.1.2.3.1 and Section 4.1.2.3.2 the implementation of NFC ISO/IEC 14443-4 protocol commands and response is required. The so-called APDUs are described below.

In total, four APDUs need to be implemented, in order to implement the EV charging protocol to be compliant to VDE-AR-E 2532-100.

SELECT for selecting the EV charging application on the chip. This command is a subset of the functionality of the ISO/IEC 7816-4 [ref.[3]] command ISOSelectFile.

ReadData for reading out the card´s certificate and potentially additional card holder data, which was injected earlier.

ECDSASign for executing the unilateral card authentication.

WriteData for writing additional information to the chip into the additional file, for example, during in the field enrolling or personalization. With this command, this file can then also be locked to avoid any further writing to it.

4.2 Benefits and security strengths of VDE-AR-E 2532-100 application rule

VDE-AR-E 2532-100 is designed to be a simple, cost-effective guideline, helping EV charging ecosystems players to transition to a secure implementation of EV charging user authorization.

- **Compatibility to existing EV charging infrastructure**
  One major benefit of the mentioned application rule is the backward compatibility of already installed EV charging infrastructures and systems. The established business logic, end user handling and concepts like "Allowlisting", "Denylisting", "Offline Operation", "Online Operation", "EV Roaming", and more are well covered by the concept, without modification of the already deployed hardware or business logic.
  The upgrade to VDE-AR-E 2532-100 involves an extension of firmware which is running on the EV charging station, but no modification or extension of hardware, so the charging equipment’s bill of materials can remain the same.
  The software upgrade requires the EV charging station to be able to handle asymmetric cryptography, public keys and certificates, which is required for reading out the smart card´s dynamic signature and its validation.
  The upgrade can be implemented in a “backward-compatible” mode, so that charge points can continue to accept existing cards, which have been issued earlier and still rely on reading out the UID only, whereas only new EV charging cards which support VDE-AR-E 2532-100 would then be accessed using the secure authorization mechanism.
  What’s more, the transition to VDE-AR-E 2531-100 can be gradual, so EV charging suppliers can plan their rollouts in the way that makes the most sense for them, offering the highest possible flexibility.
- **Relying on the UID to identify the EV charging user account**
  By still relying on the UID as the account identifier of the EV driver, the overall system architecture and user management of the eMSP and CPO don´t need to be modified and changed. The management of the backend system can remain as it is, bringing huge benefits.

After integrating the secure user authentication and authorization via NFC, the smart card`s UID can be forwarded to the backend system and used as user account identifier without problem - like it is already common practice today.
This allows uninterrupted system functionality and zero modification of the system logic.

- **No hardware modification on already deployed EV charging equipment and readers**
The concept relies on software updates only, therefore already deployed EV charging stations and equipment don´t need to exchange any installed hardware. There is no additional hardware-backed keystore required, as only asymmetric public keys are used, which don´t need to be stored on secure hardware, but can be handled in software without any problem.

5 EV charging compliant to VDE-AR-E 2532-100 with MIFARE DUOX

The MIFARE DUOX (MF3E(H)x3) smart card IC supports the command set and pre-configuration defined by standardization committee VDE-DKE in VDE-AR-E 2532-100 "Requirements for an authentication for the use of electric mobility supply systems" [ref.[4]].

NXP as a semiconductor manufacturer acts as the trusted party to do chip-individual pre-provisioning of the required data structures and key material, into every smart card IC. NXP uses its "Trust Provisioning" concept for securely injecting data and key material, as well as related certificates into each individual IC.

At the same time, NXP acts as the EV charging Certificate Authority for VDE-AR 2532-100 compliant EV-Charging certificate issuance.

MIFARE DUOX is available for sales in different product types (product configuration options):

- MIFARE DUOX standard product (with empty user memory and no pre-provisioned data)
  - Further description of this product as well as ordering information is available in the MIFARE DUOX product data sheet [ref.[5]].
- MIFARE DUOX for EV charging (with already pre-provisioned data, keys and certificate)
  - Ensures compliance to VDE-AR-E 2532-100
  - Configuration details, settings, etc., are described in detail within this document, in Section 5.1, as well as in the MIFARE DUOX product data sheet, chapter "EV Charging" [ref.[5]].
  - Ordering information for this product is available in Section 5.2.

Within this document, the focus is put on the "MIFARE DOUX for EV charging" product type.

5.1 MIFARE DUOX for EV charging - Configuration, settings, pre-personalization

The EV charging functionality, as required by VDE-AR-E 2532-100 is enabled and pre-installed on "MIFARE DUOX for EV charging" out of the box.

There is one application with AID 0xA000000845000000000000000000001“ pre-created on the IC, and the four EV charging commands which are specified by the regulation, as illustrated in Section 4.1.2.3.3, are implemented and supported by MIFARE DUOX. These commands are explained for the MIFARE DUOX context in xx.

To execute the four EV Charging commands specified successfully, the following data structures are present in the EV charging application:

- A KeyID.ECCPrivateKey with KeyNo 0x00. VDE-AR-E 2532-100 requires the key to be on brainpoolP256r1.
- A FileType.StandardData file with FileNo 0x00, holding the certificate.
- A FileType.StandardData file with FileNo 0x01, holding additional data. Here MIFARE DUOX is agnostic about the actual content.

In the following chapters, Section 5.1.2, Section 5.1.3, Section 5.1.4, Section 5.1.5, the "MIFARE DUOX for EV charging" product configuration is explained in detail.

For further technical explanation and product configuration settings and related details refer to the full MIFARE DUOX product data sheet [ref.[5]].

5.1.1 EV charging commands

The EV Charging commands as required for VDE-AR-E 2532-100 and defined in this section are ISO/IEC 7816-4 APDU from the proprietary class, that is, they have their CLA byte set to 0x80. This is different from the interindustry commands with CLA byte 0x00.

As mentioned in Section 4.1.2.3.3, four APDUs need to be supported by the smart card IC, in order to be compliant to VDE-AR-E 2532-100.

The commands are specified in detail in the MIFARE DUOX product data sheet, ref.[5], and in the appendix of this document, Section 8.

- **SELECT** for selecting the EV charging application on the chip. This command is a subset of the functionality of the ISO/IEC 7816-4 [ref.[3]] command ISOSelectFile.
  - This is the command **ISOSelectFile** on MIFARE DUOX.
  - ISOSelectFile selects the EV charging application, if the parameter P1 is set to 0x04 and the AID 0xA0000008450000000000000000000001 is provided as input parameter.
- **ReadData** for reading out the card´s certificate and potentially additional card holder data, which was injected earlier.
  - This is the command **VDE_ReadData** on MIFARE DUOX.
  - **VDE_ReadData**
    - Supports retrieving the data from FileNo 0x00 or 0x01 (if FileNo 0x00 or 0x01 are configured as FileType.StdDataFile).
- **ECDSASign** for executing the unilateral card authentication.
  - This is the command **VDE_ECDSASign** on MIFARE DUOX.
  - **VDE_ECDSASign**
    - Supports signing a 32-byte challenge (random number), which is received from the reader terminal (EV charging station).
    - The command implements an ECDSA Digital Signature Generation as defined in [ref.[6]]. The supported hash function is SHA-256, as specified in NIST FIPS 180-4 [ref.[7]].
    - The key targeted VDE_ECDSASign is the KeyID.ECCPrivateKey with KeyNo 0x00. This key has ECC Sign operations enabled.
    - The input data is the 32 random challenge and the response is the 64-byte signature consisting of (r,s).
- **WriteData** for writing additional information to the chip into the additional file, for example, during in the field enrolling or personalization. With this command, this file can then also be locked to avoid any further writing to it.
  - This is the command **VDE_WriteData** on MIFARE DUOX.
  - **VDE_WriteData**
    - Supports writing data to FileNo 0x01, and eventually locking the file once all data was written.

5.1.2 EV charging application

As the EV Charging application is pre-installed on the "MIFARE DUOX for EV-Charging" product, it comes with the following configuration.

The mentioned configuration including application structure, file structure, and keys with related certificates is present in the dedicated MIFARE DUOX EV-Charging part types.

In addition to the trust-provisioned key pair and certificate, a defined number of symmetric keys are created in the application, to support additional functionality (for example, creating additional files, or writing to file 0x0 after authentication).

- AID: 0x1010F6 (MSB first representation)
- KeySett1 (AppKeySettings): 0x09, i.e.:
  - Change key access right: 0x0.
  - AppKeySettings changeable with KeyID.AppMasterKey authentication.
  - No free file creation/deletion.
  - No free file directory access.

  - KeyID.AppMasterKey changeable with KeyID.AppMasterKey authentication.
- KeySett2: 0xB5, i.e.:
  - KeyType.AES128.
  - ISO/IEC 7816-4 File Identifiers enabled.
  - KeySett3 enabled.
  - Five application keys, initialized to the default value of all zero bytes for key value and version.
- KeySett3: 0x04, i.e.:
  - Application deletion depending on PICCKeySettings.
  - Application-specific capability data enabled, i.e. for CryptoAPI configuration.
  - No KeyID.VCProximityKey.
  - No key sets.
- ISOFileID: 0x00DF.
- ISODFName: 0xA000000845000000000000000000000001
- SetConfiguration Option 0x04 SMConfigB: 0x10, i.e. enable EV Charging.
- SetConfiguration Option 0x15 CryptoAuthAC: 0x0E, i.e. free access to VDE_ECDSASign.

Note that as five KeyID.AppKeys (including the KeyID.AppMasterKey) are initialized with the default all zero-byte value at delivery, the customer shall update those keys before handing out the card to avoid abuse in the field.

5.1.3 EV charging KeyID.ECCPrivateKey entry

The EV charging application holds a KeyID.ECCPrivateKey entry.

- Unique key pair per die that can be used for EV Charging functionality.
- The private key is stored as KeyID.ECCPrivateKey KeyNo 0x00, with the following default configuration:
  - brainpoolP256r1
  - KeyPolicy 0x0010, only allowing ECC Signing. Note that for allowing VDE_ECDSASign in VCState_DFNotAuthenticated the access condition for Crypto API Management CryptoAuthAC (as in general configurable with Cmd.SetConfiguration Option 0x15) shall be set to 0xE for this application.
  - WriteAccess: 0x0F, disabling update.
  - KUCLimit: disabled.
- The public key is defined and trusted via the certificate stored in File 0x00, see Section 5.1.4 and Section 5.1.5.

5.1.4 EV charging files

The EV Charging application holds two files.

A first file holding the certificate is a FileType.StandardData file of 193 or 199 bytes (respectively for 7-byte or 10-byte UID) with the following properties:

- FileNo = 0x00; ISO File ID = 0x00EF
- FileAR.Read = 0xE; FileAR.Write = 0x0; FileAR.ReadWrite = 0x0; FileAR.Change = 0x0.
- Secure Dynamic Messaging and mirroring is not supported for this file.
- Additional access rights are not supported for this file.
- CommMode.Full.

This file holds the EV Charging Certificate as defined in Section 5.1.5.

A second file in the application is a FileType.StandardData file of 1024 bytes with the following properties:

- FileNo = 0x01; ISO File ID = 0x01EF

- FileAR.Read = 0xE; FileAR.Write = 0xE; FileAR.ReadWrite = 0xE; FileAR.Change = 0xE.
- Secure Dynamic Messaging and mirroring is not supported for this file.
- Additional access rights are not supported for this file.
- CommMode.Plain: This is not relevant, as only free access is supported.
- This file will be filled with zero bytes at delivery.

5.1.5 EV charging certificate

As defined above, file 0x00 will hold the EV charging certificate over the public key of the application's key pair, signed by a certificate authority (CA) maintained by NXP. This CA is dedicated for the EV charging application on MIFARE DUOX. As this application will be pre-installed, it means that the CA is shared across different customers. Therefore, the customer will need to allowlist instances before adopting them in their EV charging system.

The CA maintained by NXP dedicated for the EV charging application is identifiable via the CA ID 63709320010002.

Certificates of this EV charging CA can be retrieved via the following link: https://www.gp-ca.nxp.com/CA/getCA?caid=63709320010002

Note: The CA-ID might be changed in the future, so the implementation should check the CA-ID contained in the certificate accordingly.

The certificate has a GlobalPlatform structure as defined by [ref.[4]], with following specifics:

Table 1. EV charging certificate

Tag	Length	Value	Description
0x7F21	179 or 185	-	Certificate
0x93	7 or 10	UID	Certificate serial number encoding the 7-byte or 10-byte UID
0x42	7	CA-ID	CA Identifier holding BCD-encoded 14-digit CA-ID allowing retrieval from https://www.gp-ca.nxp.com/CA/getCA?caid=63709320010002
0x5720	7 or 10	UID	Subject Identifier holding the 7-byte or 10-byte UID
0x95	2	0x02 0x00	Key usage indicating digital signature
0x5F25	4	YYYMMDD (BCD format)	Effective date, holding production date
0x5F24	4	YYYMMDD (BCD format)	Expiration date, holding production date + 20 years
0x45	1	0x00	CA Security Domain Image Number, fixed to 0x00 by [ref.[4]]
0x7F49	70	-	Public key data object, see Table 2.
0x5F37	64	r || s	Signature: ECDSA with SHA-256 signature (without additional encoding). The signed data include the tags 0x93 until and including 0x7F49.

Table 2. EV charging certificate public key

Tag	Length	Value	Description
0xB0	65	0x04||Pub.x||Pub.y	ECC public key in uncompressed point representation

Table 2. EV charging certificate public key*...continued*

Tag	Length	Value	Description
0xF0	1	0x03	Key Parameter Reference: brainpool P256r1

5.2 Ordering information for MIFARE DUOX for EV charging

Table 3. Ordering information for MIFARE DUOX for EV-Charging MF3E(X)x3.../01EV

Type number	Package		
	Name	Description	Version
MF3E23A1DUF/01EV	FFC	12 inch wafer (sawn; 75 µm thickness)[1][2]; 2 kB, 17 pF input capacitance	-
MF3E43A1DUF/01EV	FFC	12 inch wafer (sawn; 75 µm thickness)[1][2]; 4 kB, 17 pF input capacitance	-
MF3E83A1DUF/01EV	FFC	12 inch wafer (sawn; 75 µm thickness)[1][2]; 8 kB, 17 pF input capacitance	-
MF3E93A1DUF/01EV	FFC	12 inch wafer (sawn; 75 µm thickness)[1][2]; 16 kB, 17 pF input capacitance	-
MF3EH23A1DUF/01EV	FFC	12 inch wafer (sawn; 75 µm thickness)[1][2]; 2 kB, 70 pF input capacitance	-
MF3EH43A1DUF/01EV	FFC	12 inch wafer (sawn; 75 µm thickness)[1][2]; 4 kB, 70 pF input capacitance	-
MF3EH83A1DUF/01EV	FFC	12 inch wafer (sawn; 75 µm thickness)[1][2]; 8 kB, 70 pF input capacitance	-
MF3EH93A1DUF/01EV	FFC	12 inch wafer (sawn; 75 µm thickness)[1][2]; 16 kB, 70 pF input capacitance	-
MF3E23A0DA8/01EV	MOA8	plastic leadless module carrier package[3]; 2 kB, 17 pF input capacitance	SOT500-4
MF3E43A0DA8/01EV	MOA8	plastic leadless module carrier package[3]; 4 kB, 17 pF input capacitance	SOT500-4
MF3E83A0DA8/01EV	MOA8	plastic leadless module carrier package[3]; 8 kB, 17 pF input capacitance	SOT500-4
MF3E93A0DA8/01EV	MOA8	plastic leadless module carrier package[3]; 16 kB, 17 pF input capacitance	SOT500-4
MF3EH23A0DA8/01EV	MOA8	plastic leadless module carrier package[3]; 2 kB, 70 pF input capacitance	SOT500-4
MF3EH43A0DA8/01EV	MOA8	plastic leadless module carrier package[3]; 4 kB, 70 pF input capacitance	SOT500-4
MF3EH83A0DA8/01EV	MOA8	plastic leadless module carrier package[3]; 8 kB, 70 pF input capacitance	SOT500-4
MF3EH93A0DA8/01EV	MOA8	plastic leadless module carrier package[3]; 16 kB, 70 pF input capacitance	SOT500-4

[1] Delivered on film frame carrier with electronic fail die marking according to SECSII format.
[2] See ref.[8].
[3] Refer to the MOA8 package outline in ref.[5].

6 NFC-based EV charging transaction compliant to VDE-AR-E 2532-100 with MIFARE DUOX

The EV charging transaction according to VDE-AR-E 2532-100 can be executed in two different ways, online and offline.

For the offline transaction, all verification steps need to be done on the charging station itself, but for the online variant, the cryptographic operations are offloaded into the backend.

Generally, the transaction consists of the following elements:

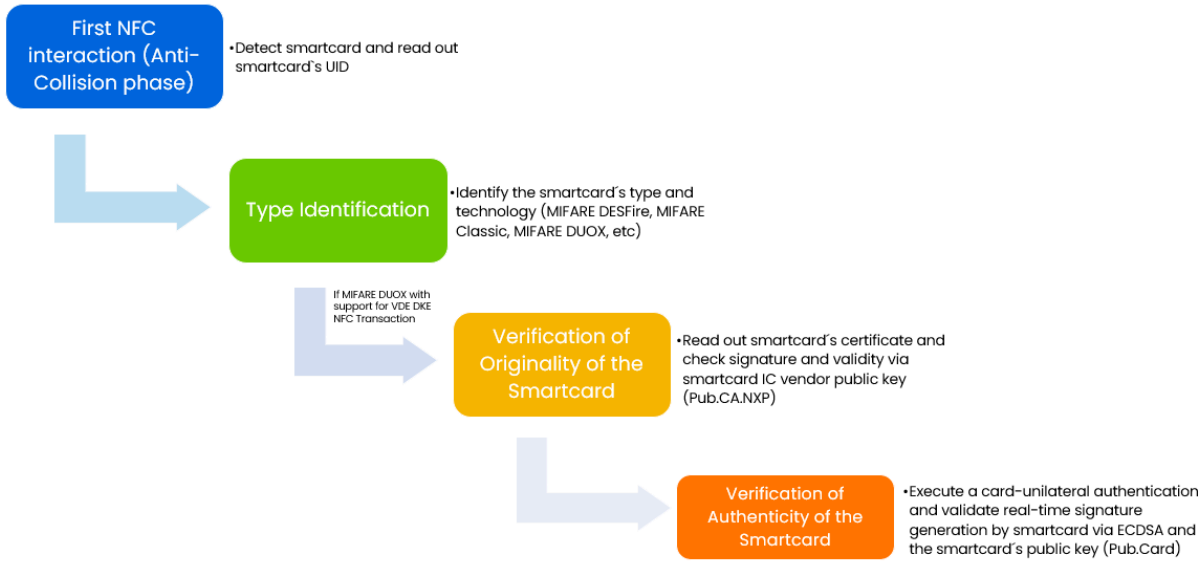


Figure 2. EV Charging transaction elements

The offline transaction looks like shown in Figure 3

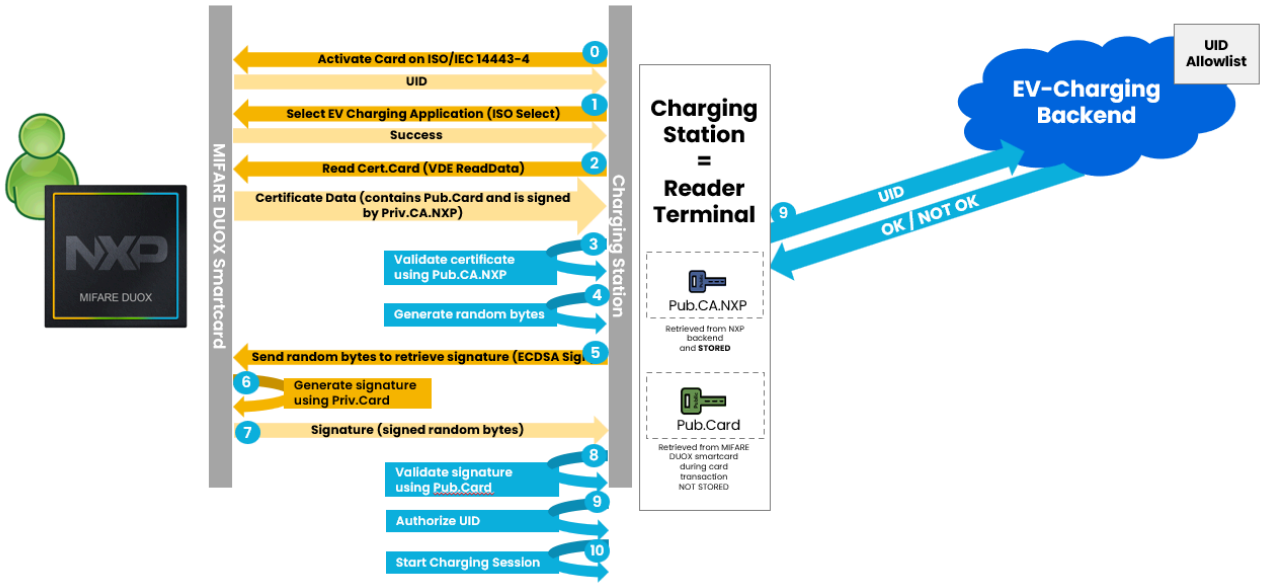


Figure 3. Offline EV Charging transaction

In this offline variant, only the last decision (step 9) is done by the backend, which is evaluating if the UID (i.e. this specific cardholder) is allowed to charge. Everything else is handled directly on the charging station.

In contrary, the online variant looks like Figure 4

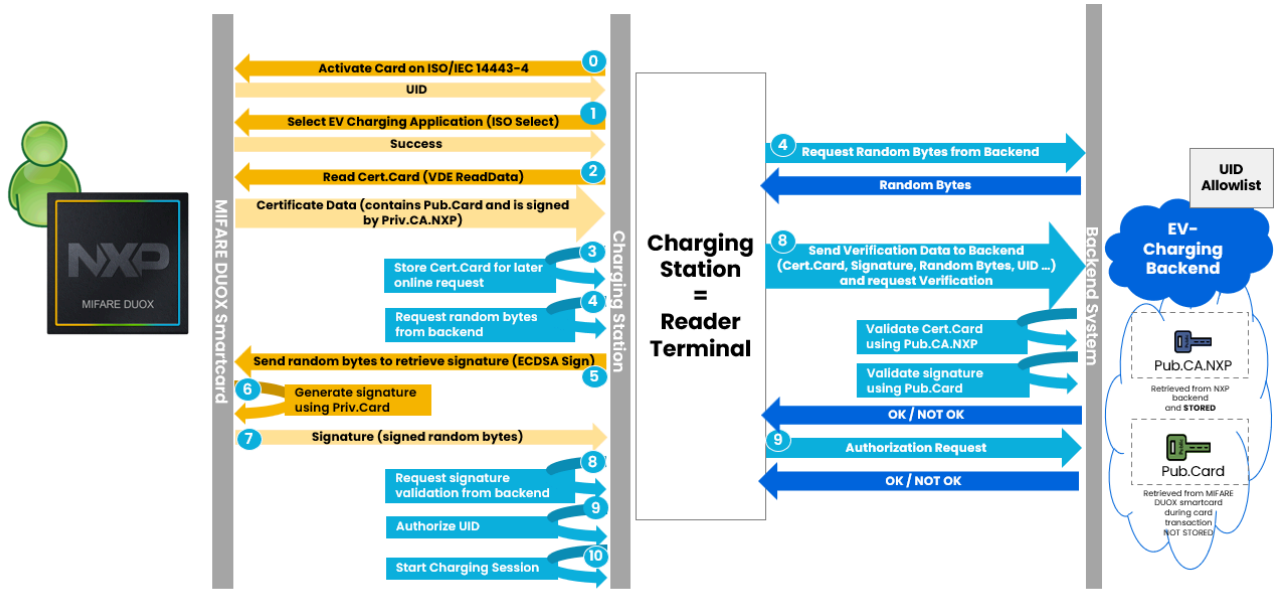


Figure 4. Online EV Charging transaction

The major difference to the offline transaction variant is that all cryptographic operations are carried out in the backend. For this, multiple backend requests are needed, which might take more time than the offline variant, but no modification on the charging station is needed.

In the following sections both variants are described further:

- Offline, with the user authentication and authorization being implemented on the charge point directly, details in Section 6.1
- Online, with the backend system being required to perform the user authentication and authorization, details in Section 6.2

6.1 Offline implementation of user authentication compliant to VDE-AR-E 2532-100

The below explained EV charging transaction using the VDE command set supported on MIFARE DUOX is designed to provide a proof that a card (respectively its UID) is not cloned or guessed, but it is a real unique serial number issued by NXP on a MIFARE DUOX card.

It also proves that the card securely authenticates itself to the reader and is a genuine product manufactured by NXP.

On a high level, this transaction reads an NXP issued certificate from the card, which signed the cards public key and its UID using the NXP owned CA root key (root of trust) and also proves that the MIFARE DUOX card indeed owns the private key linked to the signed public key of the certificate, by signing a random message sent from the reader.

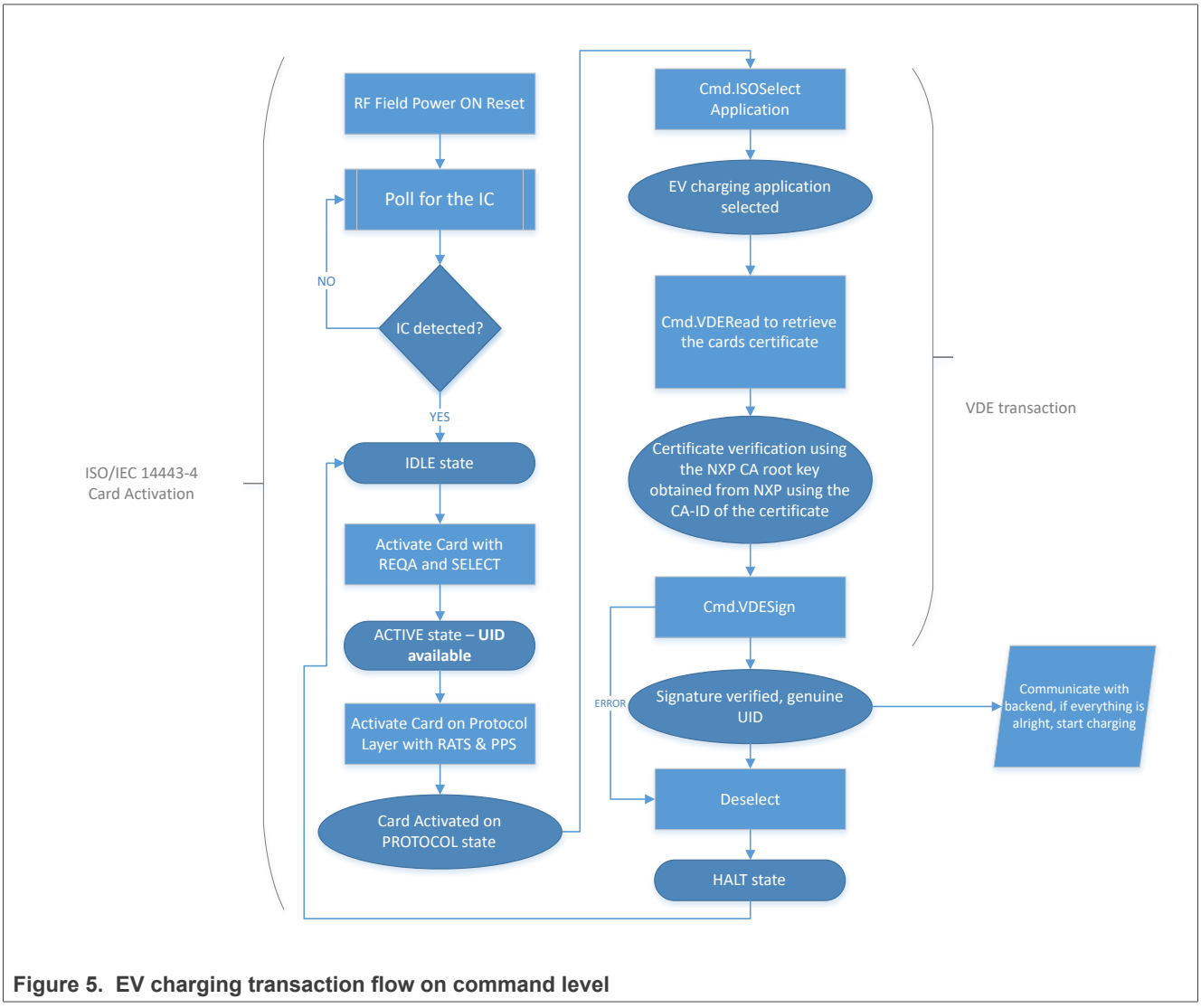


Figure 5. EV charging transaction flow on command level

Table 4. EV charging transaction

Step	Command	Direct	Data	Comment
1	IsoSelectFile	>	00A4040010A000000845000000000000000000000100	Select the EV charging application using the ISO DF Name of the application, A000000845000000000000000000000001, using Cmd.ISOSelectFile
2	Response	<	9000	
3	VdeReadData	>	8002000000	The Cmd.VDERead is used to retrieve the cards certificate. As described in Section 5.1.4, the FileNo 0x00 holds the certificate. The return value is the whole file content, respectively the whole certificate. The certificate is formatted as described in Section 5.1.5 and the return code as SW1 being 0x90 and SW2 being 0x00 meaning the command was successfully executed.
4	Response	<	7F2181BD9307042F02B1D0899042076370932001 00025F2007042F02B1D08990950202005F250420 2412135F2404204412134501007F4946B04104A7 C66CF7E84439701591AB4F7B479425C564D49 ABBAD1BC6DD83A251B0931B291FF82A0F5DC 7AC19418C2E3BB40B8E106CA8287F30AC901 D841B6A393EFE782CF001035F3740469CACCA5 0AB9277AD6C431125D9EFEE413EC23D9253DA 12356984968D09B68FA0F2BF6C22A374DBC9 DFF46CADA50EA4DA4B3387B9DCFCB82F7C51 FF891A63009000	
5	EV Charging Certificate	=	7F21 81BD 93 07 042F02B1D08990 42 07 63709320010002 5F20 07 042F02B1D08990 95 02 0200 5F25 04 20241213 5F24 04 20441213 45 01 00 7F49 46 B0 41 04A7C66CF7E84439701591AB4F7B479425 C564D49ABBAD1BC6DD83A251B0931B291FF82 A0F5DC7AC19418C2E3BB40B8E106CA8287F30 AC901D841B6A393EFE782C F0 01 03 5F37 40 469CACCA50AB9277AD6C431125D9 EFEE413EC23D9253DA12356984968D09B68FA0F2 BF6C22A374DBC9DFF46CADA50EA4DA4B3387B9 DCFCB82F7C51FF891A6300	The EV charging certificate according to Section 5.1.5 now needs to be verified. Therefore, the CA-ID of the certificate needs to be parsed out of the certificates Tag "0x42".
6	CA-ID	=	63709320010002	The CA-ID from the NXP CA certificate.
7	Certificate Download	=	The NXP root CA certificate needs to be obtained via https://www.gp-ca.nxp.com/CA/getCA?caid=63709320010002	This link does directly point to a .crt file that contains the certificate of the NXP EV Charging root CA (NXP EV Charging RootCAvE201). The public key from this certificate needs to be used to verify the MIFARE DUOX card certificate.
8	NXP root CA public key	=	045D70C68ADC4B0A6AE45B1782D6FFC6696E5 E54F4F78445E049B5B63D11F3AFAA3E851C223336 A726CE3D97D4B6D222DB87873C541A602F506C1 FF7CB227211AF	Public key extracted from 63709320010002.crt
9	Signed Message	=	9307042F02B1D089904207637093200100025F200 7042F02B1D08990950202005F2504202412135F2 404204412134501007F4946B04104A7C66CF7E8	The signature in the card certificate is calculated over the part of the certificate data starting at tag 0x93

Table 4. EV charging transaction*...continued*

Step	Command	Direct	Data	Comment
			4439701591AB4F7B479425C564D49ABBAD1BC6 DD83A251B0931B291FF82A0F5DC7AC19418C2E3 BB40B8E106CA8287F30AC901D841B6A393EFE782 CF00103	and ending with the contents of tag 0x7F49 (included). The signature is calculated using ECDSA with SHA 256 on the brainpoolP256r1 curve. **Note:** if the NXP NFC reader library (NDA version) is used, the function phalMfDuoX_VerifySDMSignature() can be used to verify the certificate signature. in the public version ph CryptoASym_ECC_Verify() may be used directly.
10	Verification result	=	PASS	The card certificate is verified and the public key out of the card certificate can be trusted. This public key needs to be used to verify the signature retrieved in the next step.
11	Vde ECDSASign	>	80030C09201211F758A55150C4B3BF386484677 FD2BBB2760C3998D3B56F1FF5B6D092FA9B00	The Cmd.VDEECDSASign is sent to the MIFARE DUOX card with a payload of 32-byte random number. The MIFARE DUOX card will sign this message with its public key and return the resulting signature.
12	Response	<	497C97860E1CD438FA4E3976071E9BE1635D879 FFA526C3157B2735B3EF8AEFD8E76D43D029783 FADA60990F599BB5620C10B46D1FFCEC4D01211 A2ADDC9229A9000	
13	Card public key from card certificate	=	04A7C66CF7E84439701591AB4F7B479425C564D49 ABBAD1BC6DD83A251B0931B291FF82A0F5DC7 AC19418C2E3BB40B8E106CA8287F30AC901D841 B6A393EFE782C	The card public key from the card certificate (step 5)
14	Signed random message	=	1211F758A55150C4B3BF386484677FD2BBB2760 C3998D3B56F1FF5B6D092FA9B	32-byte random number as sent in step 11.
15	Card signature	=	497C97860E1CD438FA4E3976071E9BE1635D879 FFA526C3157B2735B3EF8AEFD8E76D43D029783 FADA60990F599BB5620C10B46D1FFCEC4D01211 A2ADDC9229A	64-byte long signature as responded from the MIFARE DUOX card in step 12.
16	Signature verification	=	PASS	The signature is verified, meaning the UID present in the cards certificate (which is the same as the MIFARE DUOX card UID) is valid and not cloned or otherwise manipulated, and can be used to initiate a charging operation. **Note:** if the NXP NFC reader library (NDA version) is used, the function phalMfDuoX_Verify SDMSignature() can be used to verify the certificate signature. in the public version phCryptoASym_ECC_Verify() may be used directly.

The above transaction could be implemented in a way that only MIFARE DUOX cards are accepted, meaning that this flow is always executed and any other card product that does not support the VDE command set will not be accepted.

However, it might be important that also other card products that do not support the VDE-AR-E 2532-100 command set still need to be supported. In this case, the MIFARE DUOX GetVersion command can be utilized

to check if a card is a MIFARE DUOX or anything else. (If the card supports GetVersion, a version information will be returned, otherwise the command will simply fail). If a MIFARE DUOX is detected, the above transaction can be executed, for anything else, the UID can be accepted as is. In this case, the backend system needs to have stored which UIDs need to have the transaction excluded and which not. The reason for this is that an attacker could use a genuine MIFARE DUOX UID on some cloned hardware, which does not support the GetVersion command, and therefore the UID will be accepted by the charge point without executing the transaction.

In addition to the transaction described above, additional Cmd.VDERead and Cmd.VDEWrite can be executed, for example, to write arbitrary data in the second file available in the application (FileNo 0x01). This can be used to hold any other data. Note that this file is in principal completely unprotected, only supports plain communication and does not require any authentication prior to accessing it.

In case anything else is required or desired by the charging point operator, the standard MIFARE DUOX command set, authentication, and secure messaging can be used as well in the EV charging transaction. Refer to the MIFARE DUOX data sheet ref.[5], for additional functionality and command availability of the product.

6.2 Online implementation of user authentication compliant to VDE-AR-E 2532-100

VDE-AR-E 2532-100 offers the possibility to validate the originality and authenticity of the presented smart card in an online backend system.

The online backend system can be any desired system – operated by the CPO, the eMSP, the Roaming Hub, or others – depending on the system setup and dependencies of involved parties.

This allows to implement the actual smart card validation outside of the charging point (if desired), in an online system.

No implementation of smart card certificate validation in the NFC reader firmware at the Charging Stations is required.

ECC public keys are required to be used in the online system only, but not on the Charging Point. PK_{Vendor} needs to be stored in the backend system and is required for every smart card interaction, and PK_{Card} needs to be retrieved from the smart card during the interaction and passed to the online system).

The structure of the commands sent via NFC is exactly the same as in the offline case shown in Figure 5. The only difference to the offline variant is, that steps 5 to 10 and 13 to 16 from Table 4 are executed in the backend, and no ECC implementation is needed at the charging station.

The OCPP messages included in this flow are:

1. OCPP DataTransfer:getRandomNumber (OPTIONAL)
   a. For validating the authenticity of the smart card, a **random number** is required at the Charging Point, which needs to be sent to the card. The Charging Point shall request a random number at the backend system (ensures end-to-end implementation of the concept). To speed up this process, it is suggested to fetch the random number previous to the next NFC interaction. For requesting a random number from the backend system, a **DataTransfer.req**needs to be sent from the Charging Point to the backend system.

```
Request:
{
 "vendorID": "AR-E-2532-100:2020",
 "messageId": "getRandumNumber",
 "data": ""
}
Response:
{
 "status": "Accepted",
 "data": "NPVEk17XUV+6TzAUr9=LnrG9uV4SS0QWQw4tdaTflI="
}
```

2. Signature calculation
   a. Here, the Global Platform certificate format and the brainpoolP256r1 curve need to be implemented on the backend. Details can be found in the data sheet and this application note (for example,Table 4)
3. OCPP DataTransfer:SetVerificationInformation (OPTIONAL)
   a. For performing the smart card validation in the backend system, a **DataTransfer.req** needs to be sent from the Charging Point to the backend system. The following data needs to be included:

Name	Description
transactionId	An optional indication of the charging session, in case the ID is already known to the Charging Point
randomNumber	The random number which was sent to the card to generate the smartcard-unique signature
certificate	Base 64 encoded certificate that was read out of the smart card
signature	Base64 encoded signature which was computed by the smart card (based on the provided randomNumber)
additionalInfo	Additional optional information related to the authentication protocol
metadata	Base64 encoded optional additional data which can be read out from the smart card´s memory

1. a.

```
Request:
{
 “vendorID”: “AR-E-2532-100:2020”,
 “messageId”: “setVerificationInformation”,
 “data”: “{\"randomNumber\":
\"NPVKEkI7XUV+6TzAUr9OLnrG9uV4SS0QWQw4tdaTflI=\", \"certificate\":
\"fyGB6ZMCBCpCHGJpbmFyeS1idXR0ZXJmbHktZGV2ZWxvcG1lbnRfIAfu7u7u7u7ulQICAF8lDjIwMj
AwMDAzLzE4LzIwXyQOMjAyNTAwMDMvMTcvMjVFAQB/SUawQQSPUTK6U6MwmE6e30waIl7LaBr64Fqb5H
yd+CjbAdgy0JsZOxPC7N+Y5kpKCvGCmsUsdRZUuMPAUFgp8EmdsWgs8AEDXzdIMEYCIQCd3RHtUXQ7DJ
NRiZlxiQ2pLfpC0zriSRIQVjF2ibkl0gIhAJBTKmoBPv06jhIcjhs3y03oJOUnb1D8lswCL3lynxII\"
,
\"signature\":
\"MEQCIFsszCr2p0hRuNgK597oSSlmNIw6Eb8Hu5jy62weTeviAiAvJ1pct9nplZhZ98cnGr8HzorkDH
X583EIPGn6hAodTg==\"}"
}
Response:
{
 “status”: “Accepted”,
 “data”: “”
}
```

2. **Authorize.req** after the validity, originality, and authenticity of the smart card was proven

7 Fraud and attacks which are relevant within the EV charging industry

- **Cloning of cards**
  Any installation that bases its identification of the user solely on some static data that is freely accessible on an NFC card (for example, the cards UID) can easily become attacked by copying this information on another card. Though no NXP NFC or MIFARE product allows its UID to be written, there are competitor products in the market that allow a rewriting of the UID using, for example, a smartphone application, computer-based application or even specific hardware tools available via online marketplaces. The attacker now only needs the information from a card belonging to the installation, for example, by reading its UID with a smartphone, and is then able to identify as the original card holder at the charging station.
- **Guessing of card UIDs**
  NXP NFC and MIFARE UIDs are assigned using a specific scheme that is not publicly available which ensures that across all NXP NFC and MIFARE products the UIDs are unique and no duplicates occur, also not between different product families. However an attacker who has access to a few UIDs from a single production batch (i.e. a single wafer), this attacker might be able to guess other UIDs that might be on this wafer, by, for example, flipping a bit in the UID. Given the fact that the ICs used in a smart card production batch are usually coming from the same wafer, it might be that the attacker guesses a UID that is used in the same installation and therefore now can create a copy of this card with the approaches discussed above. While chances are low, and possibly a lot of trial and error is needed, this attack is still a valid scenario to consider.
- **Emulating UIDs on dedicated hacking hardware**
  Since things like the Flipper Zero or Proxmark3 NFC hacking tools are more and more common and rather easily available to the general public, attackers can also use those to mount attacks as described above. With such dedicated hardware, for example , many UIDs can be guessed and tried in a short period of time, as those devices can emulate UIDs directly, eliminating the need of writing it to a card first. Similar, those devices are also able to quickly scan a UID of an existing card (for example, by quickly placing the tool on a wallet or a persons pocket) and then emulate this UID to start charging on the cost of the victim.

All the attacks described above can be simply mitigated by **not relying on a static, non-secret piece of data**, but introducing some cryptography-based authentication that protects the identification data:

- either make the static identification data available ONLY after authentication
- or make sure that the static identification data is ONLY used after the additional authentication is completed

The proposed authentication scheme in this application note which is standardized in VDE-AR-E 2532-100 id doing the latter one, which should help to integrate this flow in existing installations, by not changing how the static identification data (i.e. the card UID) looks like, but adds an additional asymmetric card-unilateral authentication that also needs to be completed.

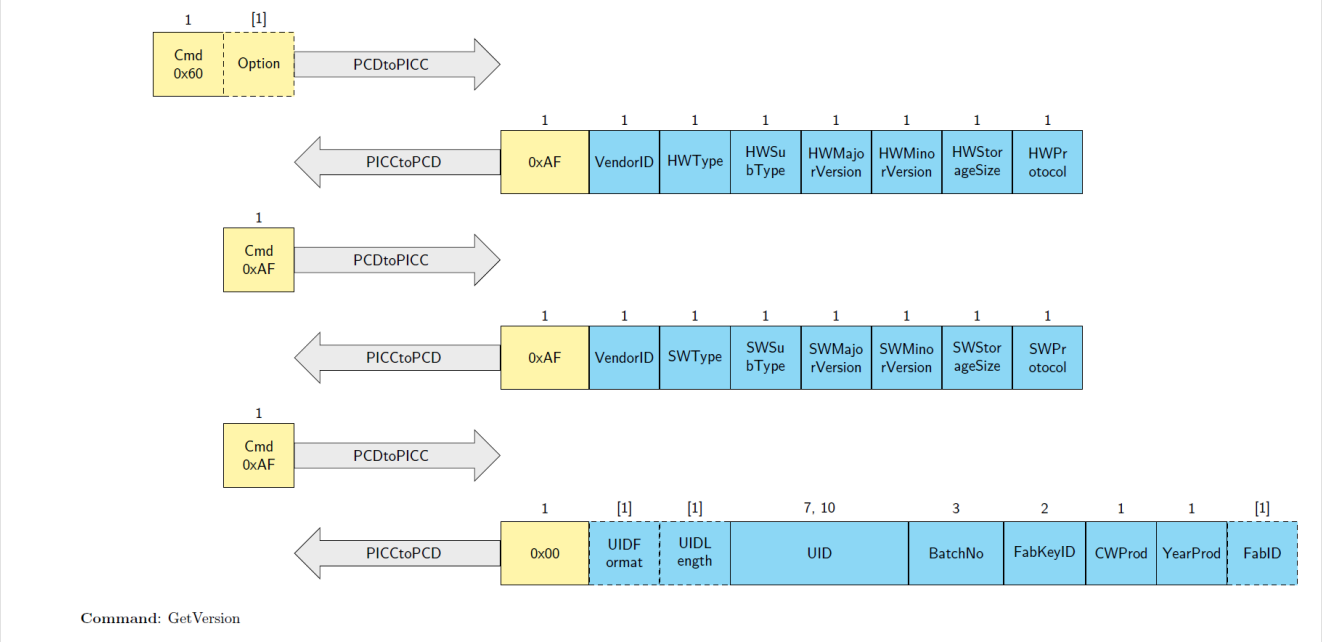
8 Appendix: MIFARE DUOX EV charging commands

A detailed description, including all supported commands, of the MIFARE DUOX and the MIFARE DUOX for EV-Charging product can be found in ref.[5].

In this section, the relevant VDE-AR-E 2532-100 compliant EV Charging commands are outlined .

8.1 GetVersion

The detailed description of this command can be found in ref.[5].



Command: GetVersion

Table 5. Command description - GetVersion

GetVersion	
Description:	Returns manufacturing related data of the PICC. First part returns HW related information.
CommMode:	CommMode.MAC

Table 6. Command parameters description - GetVersion

Name	Length	Value	Description
Command header parameters			
Cmd	1	0x60	Command code
Option	[1]	-	[Optional] Option byte
		0x01	Return Fab Identifier
Command data parameters			
-	-	-	No data parameters

Table 7. Return code description - GetVersion

Status	Value	Description
ADDITIONAL_FRAME	0xAF	
COMMAND_ABORTED	0xCA	Chained command or multiple pass command ongoing
INTEGRITY_ERROR	0x1E	Invalid secure messaging MAC (only EV2)
LENGTH_ERROR	0x7E	Command size not allowed
PARAMETER_ERROR	0x9E	Parameter value not allowed

Table 8. Response data parameters description - GetVersion - ADDITIONAL_FRAME

Name	Length	Value	Description
VendorID	1	0x04	The vendor ID
HWType	1	0x01	The HW type
HWSubType	1	-	The HW subtype
		0x01	17 pF
		0x02	70 pF
HWMajorVersion	1	0xA0	The HW major version number
HWMinorVersion	1	0x00	The HW minor version number
HWStorageSize	1	-	The HW storage size
		0x16	2 kB
		0x18	4 kB
		0x1A	8 kB
		0x1C	16 kB
HWProtocol	1	-	The HW communication protocol type
		0x05	ISO/IEC 14443-4 support
		0x20	I^2C
		0x25	I^2C and ISO/IEC 14443-4 support

Table 9. Command description - GetVersionPart2

GetVersionPart2	
Description:	Returns manufacturing related data of the PICC. Second part returns SW related information.

Table 10. Command parameters description - GetVersionPart2

Name	Length	Value	Description
Command header parameters			
Cmd	1	0xAF	Additional frame request
Command data parameters			
-	-	-	No data parameters

Table 11. Return code description - GetVersionPart2

Status	Value	Description
ADDITIONAL_FRAME	0xAF	
LENGTH_ERROR	0x7E	Command size not allowed

Table 12. Response data parameters description - GetVersionPart2 - ADDITIONAL_FRAME

Name	Length	Value	Description
VendorID	1	0x04	The vendor ID
SWType	1	0x01	The SW type
SWSubType	1	-	The SW subtype
	Bit 7-4		RFU
		0000b	
	Bit 3		
		0b	RFU
		1b	Released product configuration for product types
	Bit 2		Legacy Secure Messaging
		1b	D40 and EV1 SM not supported
	Bit 1-0	11b	RFU
SWMajorVersion	1	0x00	The SW major version number
SWMinorVersion	1	-	The SW minor version number
	Bit 7-4		Minor version
		0x0	
	Bit 3-0		Sub Minor version
		0x1	Release version
SWStorageSize	1	-	The SW storage size
		0x16	2 kB
		0x18	4 kB
		0x1A	8 kB
		0x1C	16 kB
SWProtocol	1	-	The SW communication protocol type
		0x05	ISO/IEC 14443-4 support
		0x20	I^2C
		0x25	I^2C and ISO/IEC 14443-4 support

Table 13. Command description - GetVersionPart3

GetVersionPart3	
Description:	Returns manufacturing related data of the PICC. Third part returns production related information.

Table 14. Command parameters description - GetVersionPart3

Name	Length	Value	Description
Command header parameters			
Cmd	1	0xAF	Additional frame request
Command data parameters			
-	-	-	No data parameters

Table 15. Return code description - GetVersionPart3

Status	Value	Description
OPERATION_OK	0x00	
LENGTH_ERROR	0x7E	Command size not allowed

Table 16. Response data parameters description - GetVersionPart3 - OPERATION_OK

Name	Length	Value	Description
UIDFormat	[1]	0x00	[Optional, present for non-7-byte UID] UID Format definition
UIDLength	[1]	0x0A	[Optional, present for non-7-byte UID] UID Length
UID	7, 10	-	[non-7-byte length only allowed if preceded by UIDFormat and correct UIDLength] VCUID
		All zero	if configured for RandomID (always 7 bytes returned)
		Full range	VCUID if not configured for RandomID
BatchNo	3	Full range	FabKey server batch number
TypeID	2	Limited range	Type identifier
CWProd	1	Full range	The calender week of production in BCD coding
YearProd	1	Full range	The year of production in BCD coding
FabID	[1]	Full range	[Optional, present if Option = 0x01] Fab Identifier

8.2 ISOSelectFile

The detailed description of this command can be found in ref.[5].

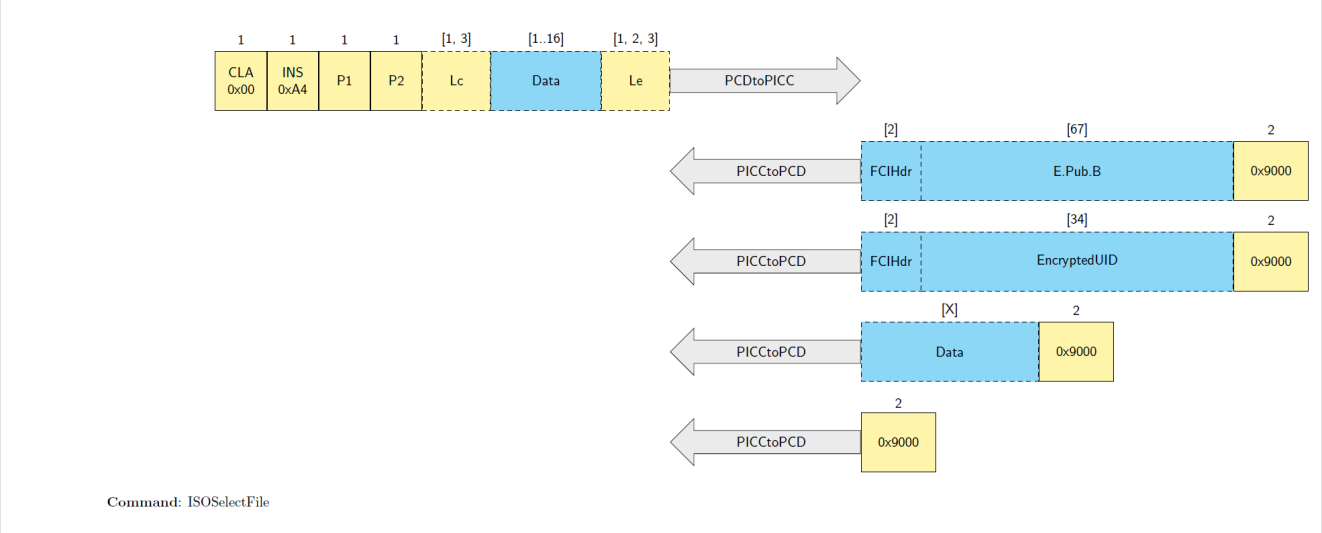


Table 17. Command description - ISOSelectFile

ISOSelectFile	
Description:	Select an application or file

Table 18. Command parameters description - ISOSelectFile

Name	Length	Value	Description
CLA	1	0x00	
INS	1	0xA4	
P1	1	-	Selection Control
		0x00	Select MF, DF or EF, by file identifier
		0x01	Select child DF
		0x02	Select EF under the current DF, by file identifier
		0x03	Select parent DF of the current DF
		0x04	Select by DF name
P2	1	-	Option
		0x00	Return FCI template
		0x0C	No response data: no FCI shall be returned
Lc	[1, 3]	0x00 .. 0x10	Length of subsequent data field
Data	[1..16]	-	Reference
		Empty	[if P1 == 0x00 OR P1 == 0x03] Select MF
		Full range	[if P1 == 0x00 OR P1 == 0x01 OR P1== 0x02] Select with the given file identifier
		Full range	[if P1 == 0x04] Select DF with the given DF name

Table 18. Command parameters description - ISOSelectFile*...continued*

Name	Length	Value	Description
Le	[1, 2, 3]	-	Empty or length of expected response
		Full range	
		0x00	Any expected length (up to 256 byte)
		0x0000	Any expected length (up to 65536 byte)
		0x45 .. 0xFFFF	[if targeted application is configured for integrated ECC-based authentication] Max expected length must be at least 69 bytes.
		0x24 .. 0xFFFF	[else if targeted application is configured for fast UID retrieval] Max expected length must be at least 36 bytes.
		0x01 .. 0xFFFF	[else] Any max expected length allowed.

Table 19. Return code description - ISOSelectFile

Status	Value	Description
ISO9000	0x9000	Correct execution, targeting application that is configured for integrated ECC-based authentication
		Correct execution, targeting application that is configured for fast UID retrieval
		Correct execution, targeting application with FCI in file ID 31
		Correct execution, no FCI requested or configured
ISO6283	0x6283	Application selected with limited functionality: TMCLimit reached
		Application selected with limited functionality: KeyID.AppTransactionSigKey does not exist or is not enabled for Transaction Signature
		Application selected with limited functionality: KeyID.AppTransactionSigKey has enabled KeyUsageCtrLimit reached
ISO6700	0x6700	Wrong or inconsistent APDU length
ISO6985	0x6985	Wrapped chained command or multiple pass command ongoing
ISO6A82	0x6A82	Application or file not found, currently selected application remains selected
ISO6A86	0x6A86	Wrong parameter P1 and/or P2
ISO6A87	0x6A87	Wrong parameter Lc inconsistent with P1-P2
ISO6C00	0x6C00	Wrong Le: expected length insufficient for response data

Table 20. Response data parameters description - ISOSelectFile - ISO9000

Name	Length	Value	Description
FCIHdr	[2]	-	[Optional] FCI Header
		T: 0x6F	Tag
		L: 0x43	Length of Value field
E.Pub.B	[67]	-	[Optional] Authentication Data Object: ephemeral public key from PICC
		T: 0x85	Tag
		L: 0x41	Length of Value field
		V: E.Pub.B	Value: ephemeral public key in uncompressed point representation (0x04 || E.Pub.B.x || E.Pub.B.y)

Table 21. Response data parameters description - ISOSelectFile - ISO9000

Name	Length	Value	Description
FCIHdr	[2]	-	[Optional] FCI Header
		T: 0x6F	Tag
		L: 0x22	Length of Value field
EncryptedUID	[34]	-	[Optional] Encrypted UID
		T: 0x85	Tag
		L: 0x20	Length of Value field
		V: RndIV || Payload	Value: - RndIV: 16 byte random - Payload: E(Kx, RndIV XOR VCData)

Table 22. Response data parameters description - ISOSelectFile - ISO9000

Name	Length	Value	Description
Data	[X]	Full Range	[Optional] FCI

Table 23. Response data parameters description - ISOSelectFile - ISO6283 - Application selected with limited functionality: TMCLimit reached

Name	Length	Value	Description
Data	[X]	Full Range	[Optional] FCI options as defined for ISO9000

Table 24. Response data parameters description - ISOSelectFile - ISO6283 - Application selected with limited functionality: KeyID.AppTransactionSigKey does not exist or is not enabled for Transaction Signature

Name	Length	Value	Description
Data	[X]	Full Range	[Optional] FCI options as defined for ISO9000

Table 25. Response data parameters description - ISOSelectFile - ISO6283 - Application selected with limited functionality: KeyIDAppTransactionSigKey has enabled KeyUsageCtrLimit reached

Name	Length	Value	Description
Data	[X]	Full Range	[Optional] FCI options as defined for ISO9000

8.3 VDE_ECDSASign

The detailed description of this command can be found in ref.[5].

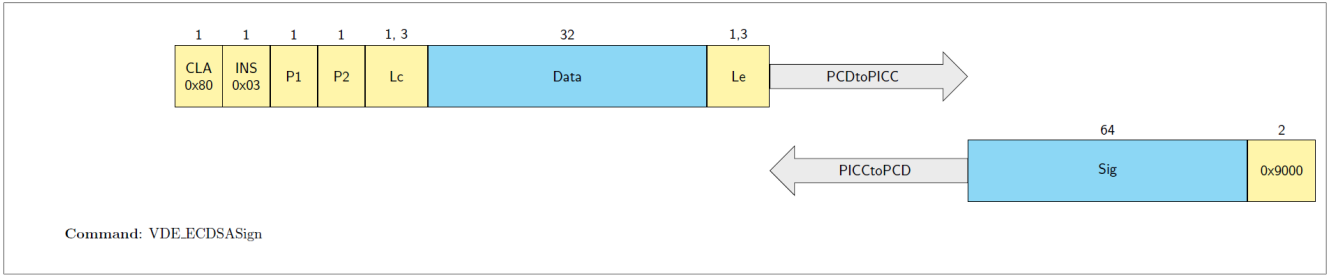


Table 26. Command description - VDE_ECDSASign

VDE_ECDSASign	
Description:	Generates and ECDSA signature over a 32-byte challenge.

Table 27. Command parameters description - VDE_ECDSASign

Name	Length	Value	Description
CLA	1	0x80	
INS	1	0x03	
P1	1	0x0C	Reserved
P2	1	0x09	Reserved
Lc	1, 3	0x20	Length of subsequent data field
Data	32	Full range	32-byte arbitrary data to sign
Le	1,3	-	Length of expected response
		0x00 / 0x000000	Any expected length up to resp. 256/65536 bytes

Table 28. Return code description - VDE_ECDSASign

Status	Value	Description
ISO9000	0x9000	Correct execution
ISO6700	0x6700	Wrong or inconsistent APDU length
ISO6982	0x6982	Security status not satisfied: not allowed
		Security status not satisfied: CryptoRequest access condition set to 0xF
		Security status not satisfied: CryptoRequest access condition not granted while different from 0xF
ISO6985	0x6985	Conditions of use not satisfied: EV Charging functionality not enabled for the currently selected application
		Conditions of use not satisfied: EV Charging functionality not suppported at PICC level
		enabled for targeted key has been reached

Table 28. Return code description - VDE_ECDSASign...*continued*

Status	Value	Description
		Conditions of use not satisfied: chained command or multiple pass command ongoing
		Conditions of use not satisfied: with KeyNo 0x00 does not exist
		Conditions of use not satisfied: with KeyNo 0x00 has not ECC Sign enabled
		Conditions of use not satisfied: ECDSA Sign disabled over NFC interface
		Conditions of use not satisfied: ECDSA Sign disabled over I^2C interface
ISO6A86	0x6A86	Wrong parameter P1: different from 0x0C
		Wrong parameter P2: different from 0x09
ISO6A87	0x6A87	Unsupported Lc
ISO6C00	0x6C00	Wrong Le: different from 0x00/0x000000

Table 29. Response data parameters description - VDE_ECDSASign - ISO9000

Name	Length	Value	Description
Sig	64	Full range	ECDSA Signature

8.4 VDE_ReadData

The detailed description of this command can be found in ref.[5].

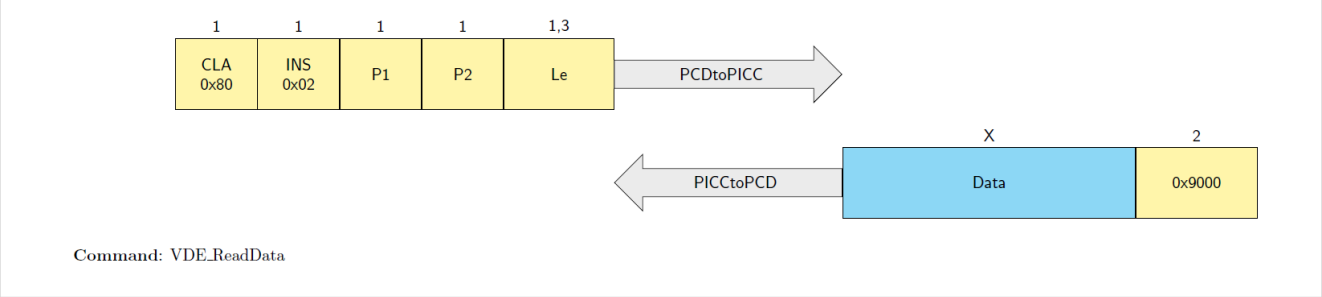


Table 30. Command description - VDE_ReadData

VDE_ReadData	
Description:	Reads data from FileType.StandardDataFile with FileNo 0x00 or 0x01.

Table 31. Command parameters description - VDE_ReadData

Name	Length	Value	Description
CLA	1	0x80	
INS	1	0x02	
P1	1	-	FileNo of the targeted file
		0x00	
		0x01	
P2	1	0x00	RFU
Le	1,3	-	Length of expected response
		0x00 / 0x000000	Any expected length up to resp. 256/65536 bytes

Table 32. Return code description - VDE_ReadData

Status	Value	Description
ISO9000	0x9000	Correct execution
ISO6700	0x6700	Wrong or inconsistent APDU length
ISO6982	0x6982	Security status not satisfied: VCState.AuthenticatedAES not allowed
		Security status not satisfied: FileAR.Read or FileAR.ReadWrite of targeted FileType.StandardDataFile file only have access conditions set to 0xF
		Security status not satisfied: FileAR.Read or FileAR.ReadWrite of targeted FileType.StandardDataFile file not granted while at least one of the access conditions is different from 0xF
ISO6985	0x6985	Conditions of use not satisfied: EV Charging functionality not enabled for the currently selected application

Table 32. Return code description - VDE_ReadData_...continued_

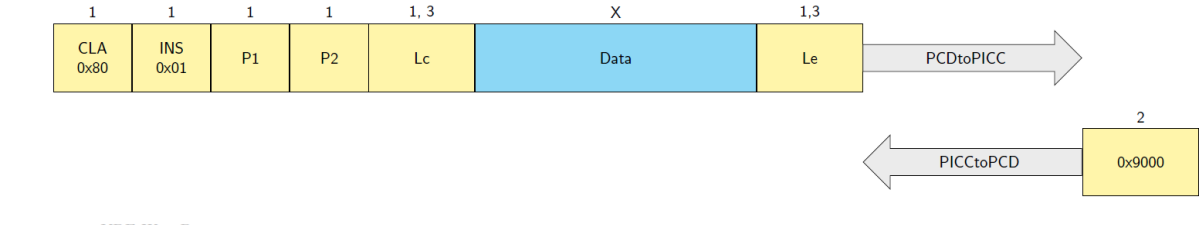
Status	Value	Description
		Conditions of use not satisfied: EV Charging functionality not suppported at PICC level
		Conditions of use not satisfied: targeted application holds a FileType.TransactionMAC
		Conditions of use not satisfied: chained command or multiple pass command ongoing
		Conditions of use not satisfied: targeted file is not of FileType.StandardData
		Conditions of use not satisfied: targeted FileType.StandardData has SDM enable
		Conditions of use not satisfied: targeted FileType.StandardData is a CRLFile
ISO6A82	0x6A82	File or application not found: targeted file does not exist in the targeted application
ISO6A86	0x6A86	Wrong parameter P1: different from 0x00 or 0x01
		Wrong parameter P2: different from 0x00
ISO6C00	0x6C00	Wrong Le: different from 0x00/0x000000

Table 33. Response data parameters description - VDE_ReadData - ISO9000

Name	Length	Value	Description
Data	X	Full range	Data read

8.5 VDE_WriteData

The detailed description of this command can be found in ref.[5].



Command: VDE_WriteData

Table 34. Command description - VDE_WriteData

VDE_WriteData	
Description:	Writes data to FileType.StandardDataFile with FileNo 0x01, and eventually lock the file.

Table 35. Command parameters description - VDE_WriteData

Name	Length	Value	Description
CLA	1	0x80	
INS	1	0x01	
P1	1	0x06	Reserved
P2	1	-	Operation
		0x00	Write
		0x01	Lock
Lc	1, 3	0x000001 .. FileSize	Length of subsequent data field
Data	X	Full range	Data to be written
Le	1,3	-	Length of expected response
		0x00 / 0x000000	Any expected length up to resp. 256/65536 bytes

Table 36. Return code description - VDE_WriteData

Status	Value	Description
ISO9000	0x9000	Correct execution
ISO6700	0x6700	Wrong or inconsistent APDU length
		Data field present while executing Lock operation
ISO6982	0x6982	Security status not satisfied: VCState.AuthenticatedAES not allowed
		Security status not satisfied: write operation (P2=0x00), and FileAR.Write or FileAR.ReadWrite of targeted FileType.StandardDataFile file only have access conditions set to 0xF

Table 36. Return code description - VDE_WriteData*...continued*

Status	Value	Description
		Security status not satisfied: lock operation (P2=0x01), and FileAR.Change of targeted FileType.StandardDataFile file only have access conditions set to 0xF
		Security status not satisfied: write operation (P2=0x00), and FileAR.Write or FileAR.ReadWrite of targeted FileType.StandardDataFile file not granted while at least one of the access conditions is different from 0xF
		Security status not satisfied: lock operation (P2=0x01), and FileAR.Change of targeted FileType.StandardDataFile file not granted while different from 0xF
ISO6985	0x6985	Conditions of use not satisfied: EV Charging functionality not enabled for the currently selected application
		Conditions of use not satisfied: EV Charging functionality not suppported at PICC level
		Conditions of use not satisfied: targeted application holds a FileType.TransactionMAC
		Conditions of use not satisfied: chained command or multiple pass command ongoing
		Conditions of use not satisfied: targeted file is not of FileType.StandardDataFile
		Conditions of use not satisfied: targeted FileType.StandardDataFile has SDM enabled
		Conditions of use not satisfied: targeted FileType.StandardDataFile is a CRLFile
		Conditions of use not satisfied: attempt to write beyond the file boundary
ISO6A82	0x6A82	File or application not found: Targeted file does not exist in the targeted application
ISO6A86	0x6A86	Wrong parameter P1: different from 0x06
		Wrong parameter P2: different from 0x00 or 0x01
ISO6C00	0x6C00	Wrong Le: different from 0x00/0x000000

9 Abbreviations

Table 37. Abbreviations

Acronym	Description
AES	Advanced Encryption Standard
AID	Application IDentifier / Application IDentification
APDU	Application Protocol Data Unit
CA	Certificate Authority
Card	A Proximity Device (PD) in a card form factor. Also called PICC.
CID	Card IDentifier
CMAC	Cipher-based MAC
Cmd	Command
CPO	Charge Point Operator
CRC	Cyclic Redundancy Check
DF	Dedicated File
ECC	Elliptic Curve Cryptography
EF	Elementary File
eMSP	Electro Mobility Service Provider
EV	Electric Vehicle
EVSE	Electric Vehicle Supply Equipment
IC	Integrated Circuit (chip)
IC Manufacturer / IC Vendor	The IC Manufacturer takes care about production of the silicon and testing the device together with initializing it with manufacturer data.
IID	Installation IDentifier
ISO Layer 3	It refers to the ISO/IEC 14443 Layer 3
ISO Layer 4	It refers to the ISO/IEC 14443 Layer 4
MF	Master File
NFC	Near Field Communication
PCD	Proximity Coupling Device (contactless reader)
PD	Proximity Device. Term used to refer to PICC or NFC device in card mode.
Personalization	The process to bind the PICC to a user, for example, storing person-related information on a PICC used for access management.
PICC	Proximity Integrated Circuit Card (Contactless Card)
PKI	Public Key Infrastructure
Pre-Personalization	The process to put keys and other Service Operator related data on a PICC at a moment that the user information is not yet known. For example, configuring PICCs before shipping them to the sales office of the buyer, where it will then be distributed further to customers, or still be used in the personalization process, if the buyer writes additional user-specific data on the IC. Pre-Personalization in the EV Charging context means configuring the PICC to be VDE-AR-E 2532-100 compliant with required PICC settings, application layout, key and certificate details.
Priv	Private Key

Table 37. Abbreviations...*continued*

Acronym	Description
Pub	Public Key
Reader Device	Reader device or reader is often used to indicate the entire PCD (= Terminal), even though the device also carries out write operations.
RF	Radio Frequency
RFID	Radio Frequency IDentification
SAM	Secure Access Module
SE	Secure Element
UID	Unique Identifier

10 References

[1] ISO/IEC 14443-3:2018 Identification cards – Contactless integrated circuit(s) cards – Proximity cards – Part 3: Initialization and anticollision – Third edition, 07 2018.

[2] ISO/IEC 14443-4:2018 Identification cards – Contactless integrated circuit(s) cards – Proximity cards – Part 4: Transmission protocol – Fourth edition, 07 2018.

[3] ISO JTC 1/SC 17 - Identification cards – Integrated circuit cards – Part 4: Organization, security and commands for interchange. ISO/IEC 7816-4:2020, May 2020.

[4] Deutsche Kommission Elektrotechnik Elektronik Informationstechnik (VDE-DKE). VDE-AR-E 2532-100 Anwendungsregel: Requirements for an authentication for the use of electric mobility supply systems, September 2020, available on dke.de.

[5] NXP Semiconductors, Datasheet, DS9744 MF3E(H)x3 MIFARE DUOX contactless smartcard IC, November 2024, available on NXP.com Secure Files.

[6] Certicom Research, Sec 1: Elliptic curve cryptography, Version 2.0, May 2009.

[7] National Institute of Standards and Technology (NIST) - Federal Information Processing Standard (FIPS) 180-4: Secure Hash Standard (SHS). NIST FIPS PUB 180-4, August 2015.

[8] NXP Semiconductors, Datasheet Addendum, AD9747: Wafer and Delivery Specification

11 Revision history

Table 38. Revision history

Document ID	Release date	Description
AN14223 v.1.0	10 July 2025	• Initial version.

Legal information

Definitions

Draft — A draft status on a document indicates that the content is still under internal review and subject to formal approval, which may result in modifications or additions. NXP Semiconductors does not give any representations or warranties as to the accuracy or completeness of information included in a draft version of a document and shall have no liability for the consequences of use of such information.

Disclaimers

Limited warranty and liability — Information in this document is believed to be accurate and reliable. However, NXP Semiconductors does not give any representations or warranties, expressed or implied, as to the accuracy or completeness of such information and shall have no liability for the consequences of use of such information. NXP Semiconductors takes no responsibility for the content in this document if provided by an information source outside of NXP Semiconductors.

In no event shall NXP Semiconductors be liable for any indirect, incidental, punitive, special or consequential damages (including - without limitation - lost profits, lost savings, business interruption, costs related to the removal or replacement of any products or rework charges) whether or not such damages are based on tort (including negligence), warranty, breach of contract or any other legal theory.

Notwithstanding any damages that customer might incur for any reason whatsoever, NXP Semiconductors' aggregate and cumulative liability towards customer for the products described herein shall be limited in accordance with the Terms and conditions of commercial sale of NXP Semiconductors.

Right to make changes — NXP Semiconductors reserves the right to make changes to information published in this document, including without limitation specifications and product descriptions, at any time and without notice. This document supersedes and replaces all information supplied prior to the publication hereof.

Suitability for use — NXP Semiconductors products are not designed, authorized or warranted to be suitable for use in life support, life-critical or safety-critical systems or equipment, nor in applications where failure or malfunction of an NXP Semiconductors product can reasonably be expected to result in personal injury, death or severe property or environmental damage. NXP Semiconductors and its suppliers accept no liability for inclusion and/or use of NXP Semiconductors products in such equipment or applications and therefore such inclusion and/or use is at the customer's own risk.

Applications — Applications that are described herein for any of these products are for illustrative purposes only. NXP Semiconductors makes no representation or warranty that such applications will be suitable for the specified use without further testing or modification.

Customers are responsible for the design and operation of their applications and products using NXP Semiconductors products, and NXP Semiconductors accepts no liability for any assistance with applications or customer product design. It is customer's sole responsibility to determine whether the NXP Semiconductors product is suitable and fit for the customer's applications and products planned, as well as for the planned application and use of customer's third party customer(s). Customers should provide appropriate design and operating safeguards to minimize the risks associated with their applications and products.

NXP Semiconductors does not accept any liability related to any default, damage, costs or problem which is based on any weakness or default in the customer's applications or products, or the application or use by customer's third party customer(s). Customer is responsible for doing all necessary testing for the customer's applications and products using NXP Semiconductors products in order to avoid a default of the applications and the products or of the application or use by customer's third party customer(s). NXP does not accept any liability in this respect.

Terms and conditions of commercial sale — NXP Semiconductors products are sold subject to the general terms and conditions of commercial sale, as published at https://www.nxp.com/profile/terms, unless otherwise agreed in a valid written individual agreement. In case an individual agreement is concluded only the terms and conditions of the respective agreement shall apply. NXP Semiconductors hereby expressly objects to applying the customer's general terms and conditions with regard to the purchase of NXP Semiconductors products by customer.

Export control — This document as well as the item(s) described herein may be subject to export control regulations. Export might require a prior authorization from competent authorities.

Suitability for use in non-automotive qualified products — Unless this document expressly states that this specific NXP Semiconductors product is automotive qualified, the product is not suitable for automotive use. It is neither qualified nor tested in accordance with automotive testing or application requirements. NXP Semiconductors accepts no liability for inclusion and/or use of non-automotive qualified products in automotive equipment or applications.

In the event that customer uses the product for design-in and use in automotive applications to automotive specifications and standards, customer (a) shall use the product without NXP Semiconductors' warranty of the product for such automotive applications, use and specifications, and (b) whenever customer uses the product for automotive applications beyond NXP Semiconductors' specifications such use shall be solely at customer's own risk, and (c) customer fully indemnifies NXP Semiconductors for any liability, damages or failed product claims resulting from customer design and use of the product for automotive applications beyond NXP Semiconductors' standard warranty and NXP Semiconductors' product specifications.

HTML publications — An HTML version, if available, of this document is provided as a courtesy. Definitive information is contained in the applicable document in PDF format. If there is a discrepancy between the HTML document and the PDF document, the PDF document has priority.

Translations — A non-English (translated) version of a document, including the legal information in that document, is for reference only. The English version shall prevail in case of any discrepancy between the translated and English versions.

Security — Customer understands that all NXP products may be subject to unidentified vulnerabilities or may support established security standards or specifications with known limitations. Customer is responsible for the design and operation of its applications and products throughout their lifecycles to reduce the effect of these vulnerabilities on customer's applications and products. Customer's responsibility also extends to other open and/or proprietary technologies supported by NXP products for use in customer's applications. NXP accepts no liability for any vulnerability. Customer should regularly check security updates from NXP and follow up appropriately.

Customer shall select products with security features that best meet rules, regulations, and standards of the intended application and make the ultimate design decisions regarding its products and is solely responsible for compliance with all legal, regulatory, and security related requirements concerning its products, regardless of any information or support that may be provided by NXP.

NXP has a Product Security Incident Response Team (PSIRT) (reachable at PSIRT@nxp.com) that manages the investigation, reporting, and solution release to security vulnerabilities of NXP products.

NXP B.V. — NXP B.V. is not an operating company and it does not distribute or sell products.

Trademarks

Notice: All referenced brands, product names, service names, and trademarks are the property of their respective owners.

NXP — wordmark and logo are trademarks of NXP B.V.

MIFARE — is a trademark of NXP B.V.

Tables

Figures

Contents