Diplomarbeit
im Diplomstudiengang Informatik
an der Universität Hildesheim

# Smartcard-Supported Internet Payments

cand. inform. Marcus Sasse

April 1998

UNIVERSITÄT HILDESHEIM

IBM

*Advisors:*

Prof. Dr. Birgit Pfitzmann (Universität des Saarlandes)
Dr. Michael Waidner (IBM Zurich Research Laboratory)
Els Van Herreweghen (IBM Zurich Research Laboratory)

UNIVERSITÄT HILDESHEIM

Institut für Informatik
Universität Hildesheim
Marienburger Platz 22
31141 Hildesheim
Germany

IBM

IBM Zurich Research Laboratory
Säumerstr. 4
8803 Rüschlikon
Switzerland

**Erklärung gemäß Diplomprüfungsordnung §18 (7):**

Hiermit versichere ich, daß ich meine Arbeit selbstständig verfaßt und keine anderen als die angegebenen Quellen und Hilfsmittel benutzt habe.

Zürich, April 1998

Marcus Sasse

# Abstract

This work analyzes the use of smartcards for the Internet payment system SET.

The security of Internet payments is one of the most important conditions for a successful realization of electronic commerce. Originally Internet payment systems such as SET were intended to perform their tasks without specific hardware support. However such software-only solutions involve security risks because they do not adequately protect the customer's data. In this work it is analyzed how the security of payment systems can be guaranteed with the additional use of trusted devices, for instance smartcards. If these trusted devices are also small enough to be carried on one's person, in addition customer mobility would also be achieved.

On such trusted and mobile devices, however, the complete SET payment system cannot be implemented because it requires the resources of a more powerful computer system. As a result one must split the payment system. In this work it is analyzed which parts can be implemented on an untrusted computer system and which parts or new tasks must be assumed by a trusted device in order to guarantee the security of the entire system. After an introduction to the SET payment system and an overview of trusted devices, attacks are described that lead to security risks. Methods are described which can be implemented on the trusted devices, and which each guarantees a certain security property of the complete system. These methods can be combined according to the possibilities of different trusted devices. In summary two implementation proposals are presented for two scenarios.

# Kurzfassung

Diese Arbeit untersucht den Einsatz von Chipkarten für das Internet-Zahlungssystem SET.

Die Sicherheit von Internet-Zahlungssystemen ist eine der wichtigsten Voraussetzungen für die erfolgreiche Realisierung des elektronischen Handels. Ursprünglich sollten Internet-Zahlungssysteme wie z.B. SET ihre Aufgabe ohne spezielle Hardwareunterstützung erfüllen. Solche reinen Softwarelösungen beinhalten jedoch Sicherheitsrisiken, da sie die Kundendaten nicht ausreichend schützen. In dieser Arbeit wird untersucht, wie durch den zusätzlichen Einsatz von vertrauenswürdigen Geräten, z.B. Chipkarten, die Sicherheit von Zahlungssystemen gewährleistet werden kann. Wenn diese Geräte zudem klein genug sind, um sie bei sich tragen zu können, wird damit gleichzeitig Kundenmobilität erzielt.

Auf solchen vertrauenswürdigen und mobilen Geräten kann jedoch nicht das gesamte SET-Zahlungssystem implementiert werden, da dieses die Ressourcen eines leistungsstärkeren Computersystems erfordert. Deshalb ist eine Teilung des Zahlungssystems notwendig. In dieser Arbeit wird untersucht, welche Teile auf einem nicht vertrauenswürdigen Computersystem implementiert werden können und welche Teile bzw. neuen Aufgaben von einem vertrauenswürdigen Gerät übernommen werden müssen, um die Sicherheit des Gesamtsystems zu garantieren. Nach einer Einführung in das SET-Zahlungssystem und einem Überblick über vertrauenswürdige Geräte werden Angriffsmöglichkeiten beschrieben, die ein Sicherheitsrisiko darstellen. Danach werden Methoden vorgestellt, die auf den vertrauenswürdigen Geräten ausgeführt werden können und jeweils einzelne Sicherheitseigenschaften des Gesamtsystems garantieren. Diese Methoden lassen sich je nach Leistungsstärke der vertrauenswürdigen Geräte bedarfsgerecht kombinieren. Zusammenfassend werden Implementierungsvorschläge für zwei ausgewählte Szenarien vorgestellt.

# Table of Contents

## Table of Attacks (Section 4.3)



## Table of Methods (Section 4.4)

## Table of Figures

# Table of Abbreviations

| | |
|---|---|
| **ASN.1** | Abstract Syntax Notation One |
| **AuthAttr** | Authenticated Attributes |
| **BCA** | Brand Certificate Authority |
| **BIN** | Bank Identification Number |
| **CA** | Certificate Authority |
| **CBC** | Cipher Block Chaining |
| **CCA** | Customer Certificate Authority |
| **CPU** | Central Processing Unit |
| **CRL** | Certificate Revocation List |
| **Cert-PE** | Certificate of the Payment Gateway’s Public Encryption Key |
| **DD** | Digested Data |
| **DER** | Distinguished Encoding Rules |
| **DES** | Data Encryption Standard |
| **EEPROM** | Electrically Erasable Programmable Read-Only Memory |
| **EX** | Extra Encryption operator |
| **Enc** | Simple Encapsulation with signature |
| **EncB** | Simple Encapsulation with signature and encrypted baggage |
| **EncBX** | Extra Encapsulation with signature and encrypted baggage |
| **EncK** | Simple Encapsulation with signature and a provided key |
| **GCA** | Geopolitical Certificate Authority |
| **HMAC** | Keyed Hashing Message Authentication Code |
| **HTML** | HyperText Markup Language |
| **HTTP** | HyperText Transfer Protocol |
| **IEC** | International Electronical Commission |
| **ISO** | International Organisation for Standardisation |
| **L** | Linkage operator |
| **MAC** | Message Authentication Code |
| **MCA** | Merchant Certificate Authority |
| **MIME** | Multipurpose Internet Message Extensions |
| **MOTO** | Mail Order/Telephone Order |
| **OAEP** | Optimal Asymmetric Encryption Padding |
| **OD** | Order Description |
| **OI** | Order Instruction or Order Information |
| **PAN** | Private Account Number |
| **PCA** | Payment Gateway Certificate Authority |

| | |
|---|---|
| **PDA** | Personal Digital Assistant |
| **PI** | Payment Instruction or Payment Information |
| **PIN** | Personal Identification Number |
| **PKCS** | Public Key Cryptography Standards |
| **POS** | Point of Sale |
| | |
| **RAM** | Random Access Memory |
| **RCA** | Root Certificate Authority |
| **ROM** | Read-Only Memory |
| **RSA** | Rivest Shamir Adleman |
| | |
| **SET** | Secure Electronic Transaction |
| **SHA-1** | Secure Hash Algorithm |
| **SO** | Signature Only operator |

# 1. Introduction

## 1.1. Scope of this Work

### 1.1.1. Motivation

With the increasing popularity of the Internet, electronic commerce is gaining importance. In an ideal realization of electronic commerce, all steps of a purchase can be performed over the Internet. This includes the selection of articles, delivery (of course only if the articles selected are digital), and payment. Currently the latter has been realized least, mainly because of security problems in today's payment system implementations. Therefore the solution to these security problems is an essential challenge to enabling real electronic commerce.

One payment system, which is regarded to become one of the most widely established systems in the near future, is the Secure Electronic Transaction (SET) Internet payment system. This work focuses on the SET system, which is designed to achieve high security. But for its implementation, security problems remain to be solved as described in the following.

### 1.1.2. Starting Point and Goal

The SET system consists of several parts. One part has to be implemented at the customer interface. The SET system was originally designed to make the implementation of the customer part possible on ordinary personal computers or workstation systems. On ordinary workstations, several applications are executed for which the customer has no possibility to check their trustworthiness. This can lead to security problems. The introduction of viruses or Trojan horses is difficult to prevent. Moreover it is difficult to guarantee that the payment application and the customer's sensitive data are sufficiently protected. Therefore in this work the assumption is made that the workstation is untrusted.

Other hardware devices exist, however, that can be trusted more because their hardware design is specialized for security purposes and they are dedicated to a certain purpose. There is no downloading of new applications into this device and therefore no security problem caused by viruses or Trojan horses. Therefore in this work the assumption is made that these devices are trusted regarding certain purposes.

If these devices are used to secure the payment application and if they are small enough to be carried on one's person, customer mobility can be achieved as well. Therefore in the following it is assumed that the trusted devices used are mobile devices, e.g. smartcards.

The goal of this work is to enable an implementation of the SET customer part with trusted device support, especially smartcard support. Thus the security or at least particular security properties of the payment system shall be guaranteed to the customer.

### 1.1.3. Problem

An implementation of the entire SET customer part does not fit on current trusted mobile devices. Therefore it is necessary to split the system. In this work this split is done on the implementation level. The resulting implementation of the customer part is compatible with the specification of the SET system.

The specified SET system does not support special hardware. In addition no clear relation is given between various parts of the system and particular purposes. If this were the case, just those parts with security critical functions would have to be implemented on the trusted device. It is the task of this work to determine these parts.

Furthermore in the SET design the resources of a workstation are assumed. These resources are far richer than those of mobile trusted devices. As a result, several parts with security-critical functions are intended to be implemented as a whole and do not fit on mobile devices. It is the task of this work to determine how security purposes can be at least partly fulfilled in such a case and possibly which new tasks are necessary for the trusted devices to perform.

### 1.1.4. Structure

This work consists of five chapters. The first chapter is a description of this work and payment systems in general. The next two chapters are about the basis of this work, the SET payment system and trusted devices. In each of these two chapters a general overview is given and then the issues relevant to this work are described. The fourth chapter is the main chapter. Here a trusted device support for the customer part of SET is discussed and solutions are developed. The last chapter is a brief comment on open problems and prospects.

## 1.2. Different Payment Systems

This work deals with the payment system SET. In addition to this system, a variety of other payment systems exist. This section shall introduce them and clarify the position of SET. In the following, essential properties are described and existing payment systems are classified.

A common property of all payment systems is their ability to transfer a financial value from a payer to a payee. Besides the payer and the payee, one or more financial institutions are involved in a payment system to ensure that the exchanged values are linked to real financial values. In most electronic payment systems these are two banks, the issuer and the acquirer, with a relationship to the payer and the payee, respectively. Independent of the particular payment system the flow of the real financial value goes always from the payer over the financial institutions to the payee as illustrated in the following diagram.

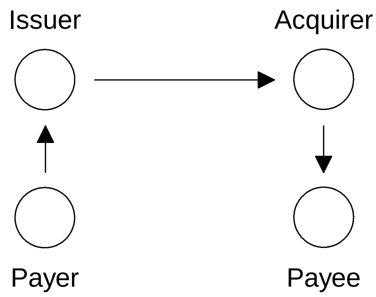


Fig. 1.1: Flow of Financial Value

Aside from this common feature, payment system properties may differ significantly. An overview of various payment systems and their properties can be found in [AJSW97]. In the following, essential properties are listed on which basis payment systems are classifiable.

*Security Properties*

Because the purpose of payment systems is to transfer financial values, great importance is attached to security properties. Particularly, if high amounts are to be transferred, security properties are essential. Different security properties are integrity/authorization, confidentiality, availability and reliability.

Payment systems differ in whether cryptography is used. Cryptography can achieve integrity/authentication with symmetric authentication or public-key digital signatures. Confidentiality can be achieved with symmetric encryption or public-key encryption. The primary purpose of public-key methods is to allow the handling of disputes between sender and recipient.

Regarding security, a further feature in which payments systems differ is the use of trusted hardware devices. At the payer's end a trusted device can be a smartcard, a mobile phone or a personal digital assistant (PDA). Also at the payee's end trusted devices are common, e.g. security modules of point-of-sale terminals or dedicated hardware for processing and storing sensitive data.

*Payment Value*

Different payment systems exist for very low-value payments and high-value payments because they place different requirements on the system. Micropayments, which are very low-value payments of less than one dollar or even fractions of a cent, require a system with low operating costs per payment. The cost per payment must not higher than the value of that payment. On the other hand payment systems for high-value payments require high security, which usually results in high costs per payment.

*Online/ Offline*

An essential property of a payment system is whether it is online or offline. Online systems require for each payment a connection between the payer and a third party (or between the payee and a third party). The third party can be a bank or a party that has a connection to a bank. Offline systems require for a payment only a connection between the payer and the payee.

*Anonymity/ Untraceability*

For the payer an important property is whether payments can be observed and stored by the payee or third parties. A system is called anonymous if the identity of the payer cannot be determined. A system is additionally called untraceable if two payments made by the same person cannot be linked to each other. Therefore systems can be divided into non-anonymous systems, anonymous but traceable systems, and anonymous as well as untraceable systems. Regarding existing systems anonymity can be subdivided further into systems where the anonymity can be destroyed by a small number of parties (e.g. anonymous remailers in Anonymous Credit Cards) and systems where anonymity cannot be destroyed (e.g. blind signatures in e-cash)

Existing payment systems are listed and divided regarding the properties online/offline and traceable/untraceable in [AJSW97]. In the following table the online and traceable systems are further separated into non-micropayment and micropayment systems.

<table>
<tr><th></th><th>online</th><th>offline</th></tr>
<tr><td rowspan="2">traceable</td><td>First Virtual, iKP, SET, CyberCash, OpenMarket</td><td rowspan="2">Danmont/Visa, CLIP, Mondex,<br>CEN Intersector Electronic Purse,<br>EMV Electronic Purse,<br>FSTC Electronic Check Project</td></tr>
<tr><td>NetBill, Millicent, μ-iKP, MiniPay</td></tr>
<tr><td>untraceable</td><td>NetCash, Phone-Ticks (CAFE),<br>Anonymous Credit Cards, e-cash</td><td>CAFE</td></tr>
</table>

*Time of Payment and Parties Involved*

Payment systems can be classified according to the time a value is taken from the payer relative to the time the payment was initiated (or the payer promised to pay the payee). Different payment models regarding this classification are described in [AASW96, p.3ff]. A similar and more detailed distinction is made in [PfWa96, p.8ff].

Payment systems are either prepaid, pay-now or pay-later systems. In prepaid systems the payer’s account must be debited before a payment can be initiated, in pay-now systems the payer's account is debited at the time of payment and in pay-later systems the payer's account is debited after the payee's account is credited. Because pay-later systems can also be used as pay-now systems and vice versa with only minor technology changes, systems can be differentiated for practical reasons in cash-like (prepaid) systems and in check-like (pay-now or pay-later) systems. Check-like systems are also called bank-card-based or account-based.

Cash-like and check-like systems are called direct systems because participation of a payer and a payee is required. If only a payer or a payee is involved (and a bank), the system is called an indirect push or indirect pull system, respectively. An example of a push system is a fixed standing order or a wire transfer; an example of a pull system is a direct debit. Home-banking systems are indirect payment systems.

<table>
<tr><td colspan="3">direct</td><td colspan="2">indirect</td></tr>
<tr><td>cash-like</td><td colspan="2">check-like/account-based</td><td rowspan="2">push</td><td rowspan="2">pull</td></tr>
<tr><td>prepaid</td><td>pay-now</td><td>pay-later</td></tr>
</table>

Direct payment systems can be online or offline. Indirect systems are always online. The existing payment systems listed above are all direct payment systems. They can also be classified on the basis of the property cash-like or check-like as shown in the following table.

| cash-like | check-like/account-based |
|---|---|
| *CEN Intersector Electronic Purse, CAFE, Phone-Ticks (CAFE), CLIP, Danmont/Visa, e-cash, EMV Electronic Purse, Millicent, Mondex, NetCash* | *Anonymous Credit Cards, CyberCash, First Virtual, FSTC Electronic Check Project, iKP, $\mu$-iKP, MiniPay, NetBill, OpenMarket, SET* |

# 2. The SET System

## 2.1. Overview

Regarding the properties given in section 1.2 the SET payment system can be classified as follows. It achieves high security particularly concerning such security aspects as integrity, authorization, confidentiality, and reliability. It is designed to run on the public network Internet and uses mainly public-key digital signatures and public-key encryption, partly symmetric encryption. SET is a system for high and low payments, but not for micropayments because the cost per payment is too high. Nevertheless it could be extended to process micropayments as well. SET is an online and traceable system. Furthermore it is a direct and account-based system (see tables in section 1.2).

Although trusted hardware devices (which are described in chapter 3) are not specified in the SET specifications [MaVi97a,b,c], they are necessary to preserve the security of the system. This is explained in section 2.2. This work deals with the addition of trusted hardware to an implementation of the system, in particular a smartcard support for the payer, and the issues concerning which parts of the system have to be executed on such a trusted device.

In the next paragraphs, properties of SET that are important for the further work are described in more detail.

### 2.1.1. Entities Involved

A payment with SET involves a collection of entities [MaVi97b, p.4,26].

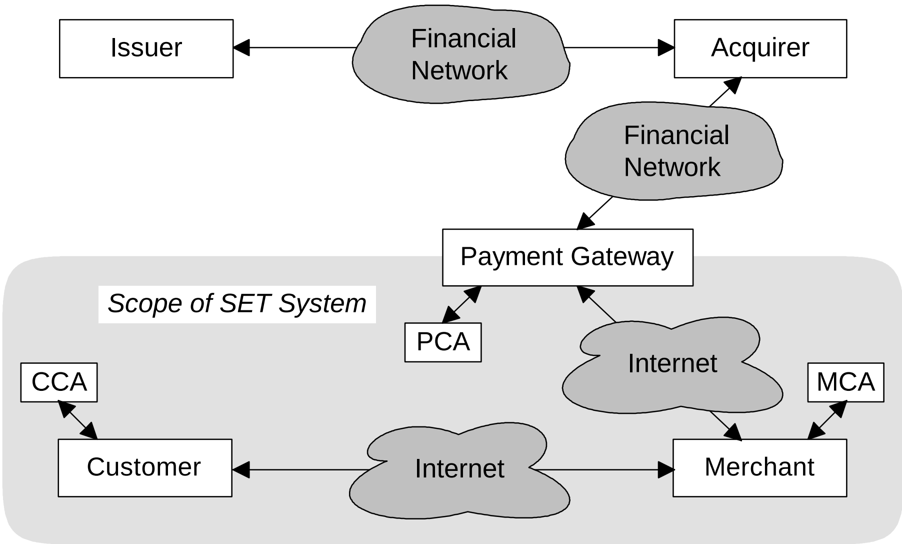


Fig. 2.1: Entities Involved

In certain payment systems the payment flow is not fixed to properties of persons: anybody can be a payer and a payee. However in SET, anybody can be a payer but only a merchant can be a payee. Therefore the payee is now called a merchant and the payer is called a customer. In the SET specification the term cardholder is used to indicate that this person is the holder of a credit card. But because with SET the possession of such a card is no longer necessary and to avoid confusion with the holder of a smartcard, the term customer is used in this work.

The financial institutions are an issuer and an acquirer. In contrast to the financial flow (cf. section 1.2) the communication in the SET system starts with a connection between the customer and the merchant. Afterwards the acquirer and then the issuer are involved. The merchant has no direct connection to the acquirer, but rather communicates with an intermediary payment gateway. This payment gateway separates the public Internet from the private financial network. It handles all messages itself on behalf of the acquirer from the SET point of view. Converting SET messages into specific messages for the financial network is the payment gateway's responsibility and out of the scope of SET. Therefore within the context of SET instead of the term payment gateway also the term acquirer is used.

SET is a three-party protocol that specifies the communication between the entities which are connected via the Internet: the customer, the merchant and the payment gateway on behalf of the acquirer. For these payment system participants, certification authority entities are necessary to enable security based on public-key cryptography. They are called the customer certification authority (CCA), the merchant certification authority (MCA), and the payment gateway certification authority (PCA). In the following paragraph the three-party communication process is described from the customer's point of view.

### 2.1.2. The Electronic Shopping Process

A shopping process consists of multiple stages. The SET protocol addresses those stages that can be carried out with the use of a conventional credit card. In the following all stages are listed and those are marked that are part of SET [MaVi97a, p.10f; MaVi97b, p. 4ff,326]. In stage 2 the data elements of the corresponding message (see paragraph 2.2.3) are given in brackets.

| | | |
|---|---|---|
| Stage 1 | - | The customer browses for items in the WWW, views a catalog on a CD-ROM or looks at a paper catalog and selects items he wants to purchase. |
| Stage 2 | - | The customer is presented with an order form containing the<br>-order description (OD) which contains the<br>-list of items and their description (quantity, size, price, etc.)<br>-shipping address, if required<br>-total price (PurchAmt) including shipping, handling and taxes<br>-installment payment data (InstallRecurData), if required<br>-extensions (ODExtensions)<br>After this stage the customer and the merchant shall have the same bit stream of this order. |
| Stage 3 | - | The customer selects the means of payment. If he chooses to pay with a credit card, he also chooses which brand he wants to use. |
| Stage 4 | SET | The customer sends a signed purchase request that contains a hash of his order form as well as further payment information to the merchant. |
| Stage 5 | SET | The merchant requests payment authorization from the cardholder’s financial institution (the issuer) via his own financial institution (the acquirer) and may send a confirmation to the customer. |
| Stage 6 | - | The merchant ships goods or performs the requested services. |
| Stage 7 | SET | The merchant requests payment from the issuer (via the acquirer). |

To the payment system SET, which performs stages 4, 5 and 7, great significance must be attached to security properties as described in section 1.2. Therefore in the next paragraph the requirements placed on the SET system are discussed. First the requirements are described that are given in the SET specifications. In the second part of paragraph 2.1.3 the requirements of iKP, a predecessor of SET, are discussed to what extent they are fulfilled by SET.

### 2.1.3. Security Requirements

#### Requirements described in the SET Specifications

The security requirements placed on the SET protocol consist essentially of confidentiality, integrity and authentication. The requirements are listed and generally described in [MaVi97a, p.4], [MaVi97a, p.6ff] and [MaVi97b, p.40ff], and collected in the table below.

Confidentiality of payment information (a1) is required, especially of account information, as well as confidentiality of order information (a2). Integrity shall be ensured for all transmitted data (b1). Furthermore the authentication of the customer, the merchant and the acquirer is required (c1, c2, c3).

The authentication of the customer (c1) shall prove that he is a legitimate user of a branded payment card account. The authentication of the merchant (c2) shall prove that he can accept branded payment card transactions through his relationship with an acquiring financial institution. The acquirer is not involved directly as an entity in the protocol, but represented by a payment gateway, which is the interface to the existing financial network. Therefore not the authentication of the acquirer, but more precisely the authentication of the payment gateway is required (c3).

The basic means used by SET are encryption to ensure confidentiality and signatures and certificates to provide integrity and authentication.

<table>
<tr><td rowspan="2">a) Confidentiality</td><td>a1) Confidentiality of Payment Information<br>(data is secret for merchant and outsiders)</td><td rowspan="2">Means:<br>Encryption</td></tr>
<tr><td>a2) Confidentiality of Order Information<br>(data is secret for payment gateway and outsiders)</td></tr>
<tr><td>b) Integrity</td><td>b1) Integrity of all Transmitted Data</td><td rowspan="4">Means: Signatures<br>and Certificates</td></tr>
<tr><td rowspan="3">c) Authentication</td><td>c1) Customer Authentication<br>(to merchant and payment gateway)</td></tr>
<tr><td>c2) Merchant Authentication<br>(to customer and payment gateway)</td></tr>
<tr><td>c3) Payment Gateway Authentication<br>(to merchant and customer)</td></tr>
</table>

Security requirements are described in more detail in [BGHH95], which is a paper about iKP, a predecessor of SET. The following part of this paragraph discusses to what extent these requirements are fulfilled by SET.

**iKP Requirements**

A starting point for the design of the SET protocol was iKP, a family of secure electronic payment protocols. In the paper of iKP [BGHH95, p.6ff] security requirements are described as requirements of the parties involved in the protocol. As in SET, the involved parties are the acquirer, the merchant and the customer. In the following it is examined to what extent the iKP requirements are also met by the SET protocol. The requirements of the acquirer, merchant and customer are marked A, M, and C, respectively. For a detailed description of the requirements see [BGHH95].

The assumption is made that all parties of the SET protocol possess their own public key pairs. This corresponds to the protocol 3KP. Hence non-repudiation is provided by SET ("Undeniable Proof" in [BGHH95]).

*A1. Proof of Transaction Authorization by Customer*

This requirement is met by the SET protocol with the use of the customer's public signature key pair and the corresponding certificates. The requirement can be divided into the following sub-requirements:

i. proof that the customer is the legitimate user of a credit card account,
ii. proof that the customer has authorized the payment transaction data,
iii. guarantee that the authorization is not replayable and not usable for other authorizations,
iv. guarantee that, even if the merchant is an adversary, a fake authorization is impossible.

Requirement i. is fulfilled by sending a signature of the customer together with the corresponding certificate to the acquirer: The signature is made with the operation SO(C, PI-TBS) in the structure PI (for the structure and data elements of SET messages see paragraph 2.2.3). The structure PI is sent together with the certificate via the merchant to the acquirer. The certificate then certifies that the customer is the legitimate user of an account number [MaVi97b, p.33]. Because the account number is obfuscated in the certificate the acquirer needs the account number in an extra data element to be able to verify the certificate. Therefore the account number is sent encrypted within the PI from the customer to the acquirer.

Requirement ii. is also fulfilled by the signature operation SO(C, PI-TBS). iKP requires that the payment transaction data consist of at least the following data [BGHH95, p.6,7]. Each of these data elements has a corresponding element in SET which is added in parentheses.

iKP payment transaction data and corresponding data in SET:
- amount (part of PurchAmt)
- currency (part of PurchAmt)
- goods description (OD, optional: ODExtensions)
- merchant identification (MerchantID)
- delivery address (part of OD, optional, [MaVi97b, p.326])
- transaction identification (TransIDs)
- time and date (PReqDate)

Each of the corresponding SET data elements is part of the structures HODInput or PIHead. These structures are signed (for a general problem of authorization and signing, cf. appendix 1).

Requirement iii. is met by SET with the use of the data XID (part of TransIDs), which uniquely identifies a single order. Because this data is also signed, an authorization is neither replayable nor usable to authorize other orders. Requirement iv. is implicitly guaranteed with the use of signatures.

The tick '√' in the following table indicates that this requirement is met by the SET protocol.

| | |
|---|---|
| A1. Requirement of Acquirer: Proof of Transaction Authorization by Customer | √ |

*A2. Proof of Transaction Authorization by Merchant*

This requirement is met by the SET protocol with the signed message AuthReq [MaVi97a, p.63]. This message is signed with the operation EncB, which includes besides encryption the signature of the entire message [MaVi97b, p.84]. With the transaction identifier XID included in the message AuthReq, the authorization is unambiguously linked to the corresponding order. Furthermore the acquirer can check whether the merchant and the customer agreed on the same content of the order. For this the acquirer can compare the hash HOIData of the order information and the hash HOD2 of order description, which are computed by the merchant and sent within the AuthReq, with the corresponding hashes computed by the customer and sent within the structure PI [MaVi97b, p.351].

| | |
|---|---|
| A2. Requirement of Acquirer: Proof of Transaction Authorization by Merchant | √ |

*M1. Proof of Transaction Authorization by Acquirer*

This requirement is met when the acquirer signs of the message AuthRes and sends it to the merchant. If the data element AuthCode in this message equals the value "approved", then the payment is authorized by the acquirer. More precisely this means that the indicated purchase amount is allowed by the issuing bank to be captured by the merchant [MaVi97b, p.356ff]. This ikP requirement can be divided into the following sub- requirements:

i. proof that the merchant is dealing with the real acquirer,
ii. proof that the acquirer has certified the authorization information.

Requirement i. is met by the use of signatures and the corresponding certificates. Requirement ii. is met by the signing of the following authorization information, which is represented in SET by the data elements added in parentheses:

- amount and currency (AuthAmt)
- time and date (PReqDate in TransIDs)
- information to identify the transaction (transaction identifier XID)

The data elements in parentheses are all included in the message AuthRes [MaVi97b, p.358ff].

| | |
|---|---|
| M1. Requirement of Merchant: Proof of Transaction Authorization by Acquirer | √ |

*M2. Proof of Transaction Authorization by Customer*

The proof of transaction authorization by the customer is given to the merchant with the message PReq. Even before the merchant receives the transaction authorization from the acquirer he has already received the message PReq from the customer. In that message the customer has authorized the acquirer to arrange that payment be transferred to the merchant (see A1.). The merchant can verify that the customer has signed the authorization of the acquirer.

| | |
|---|---|
| M2. Requirement of Merchant: Proof of Transaction Authorization by Customer | √ |

*C1. Payment without Customer's Authorization is Impossible*

This requirement is met by the SET protocol together with the fulfillment of requirement A1. The requirement can be divided into the following sub-requirements:

i. guarantee that authorization is impossible without account number and signature,
ii. guarantee that, even if the merchant is an adversary, a fake authorization is impossible,
iii. guarantee that, even if there are many previous legitimate transactions, a fake authorization is impossible.

Requirement i. is met, because SET requires the account number (PAN and CardExpiry in SET) and a customer's signature in the message PReqDualSigned. With the signature, requirement ii. is met as well. Requirement iii. is met, because no one but the customer can make the required signature, even if there are many previous signed transactions. It is a feature of the public signature algorithms that the private signature key can not be revealed even if the fraudulent person has many signatures that were made with that key.

| | |
|---|---|
| C1. Requirement of Customer: Unauthorized Payment is Impossible | √ |

*C2. Proof of Transaction Authorization by Acquirer*

This requirement is not met by the SET protocol (indicated by '-' in the following table). The transaction authorization by the acquirer would be proved if the customer could verify the signature made by the acquirer for the authorization of the payment. The information whether the acquirer has authorized the payment is sent from the merchant to the customer (in the data AuthStatus and CapStatus within the message PRes) but without a signature of the acquirer.

| | |
|---|---|
| C2. Requirement of Customer: Proof of Transaction Authorization by Acquirer | - |

*C3. Certification and Authentication of Merchant*

This requirement is met by the SET protocol with the signing of the message PRes by the merchant and the possibility to verify this signature by the customer with the corresponding certificate. The certificate, which is specific for the SET protocol, certifies to the customer that the merchant is accredited at an acquirer [MaVi97a, p.50].

| | |
|---|---|
| C3. Requirement of Customer: Certification and Authentication of Merchant | √ |

*C4. Receipt from Merchant*

The requirement can be divided into the following sub-requirements:

i. proof that the merchant received payment,
ii. proof that the merchant has promised to deliver goods.

If the merchant has received payment (i.), or more precisely, if a capture has successfully been performed, this is indicated by the following: The data element CompletionCode in PRes or InqRes is set to either the value 'capturePerformed' or 'creditPerformed' and the data element CapCode is set to 'success'.

If the CompletionCode in the PRes or InqRes is set neither to 'orderRejected' nor to 'orderNotReceived', it indicates that the purchase request from the customer has been accepted by the merchant. This implies that he has promised (ii.) to deliver goods (of course only under the condition that he will receive payment).

But there is no fair exchange of the message PReq and the response of the merchant. Moreover the merchant can send a PRes before the payment is finished. The customer does not get told when the payment will be finished and has to keep polling the merchant.

Furthermore the response message confirms only that a particular purchase request has been performed (defined with XID). The response message does not confirm the content of the purchase, because it contains neither the content itself (HODInput) nor a hash of it (HOD).

Hence the requirement is not fulfilled.

| | |
|---|---|
| C4. Requirement of Customer: Receipt from Merchant | - |

*C5. Privacy*

The SET protocol provides some privacy for several data of the customer. The payment information (PI) is kept secret from all but the customer and the acquirer. The data HODInput (which in iKP is the order information) is kept secret from all but the customer and the merchant. SET provides the privacy of these data with different methods.

Hybrid encryption is used for the structure PI. For the PANData in PI, which includes the account number and expiry date, the strongest method in SET is used for privacy, the encryption with the public key of the acquirer. The rest of the PI is encrypted with a symmetric key.

The data HODInput, which includes the order description, is kept secret in SET from the acquirer by sending only a salted hash (SHA-1 hash) of the data to the acquirer. In addition, HODInput must be transferred between the customer and the merchant by another application. Its privacy is therefore also dependent on that application.

The privacy of the order information is regarded to be not important. Its encryption is left to the underlying communication application. Since the privacy of some data is dependent on other applications, the tick below is put in parentheses.

| | |
|---|---|
| C5. Requirement of Customer: Privacy | (√) |

*C6. Anonymity*

In the SET protocol, anonymity of the customer from eavesdroppers and the merchant is supported (indicated by √ in the table below) but anonymity from the acquirer is not (indicated by -). As in iKP, anonymity from eavesdroppers and the merchant is supported in the sense that the customer uses a pseudo-identity.

| | |
|---|---|
| C6. Requirement of Customer: Anonymity | √- |

All requirements described in the iKP paper and discussed in this paragraph are listed in the following table.

| | | |
|---|---|---|
| Requirements of Acquirer | A1. Proof of Transaction Authorization by Customer | √ |
| | A2. Proof of Transaction Authorization by Merchant | √ |
| Requirements of Merchant | M1. Proof of Transaction Authorization by Acquirer | √ |
| | M2. Proof of Transaction Authorization by Customer | √ |
| Requirements of Customer | C1. Unauthorized Payment is Impossible | √ |
| | C2. Proof of Transaction Authorization by Acquirer | - |
| | C3. Certification and Authentication of Merchant | √ |
| | C4. Receipt from Merchant | - |
| | C5. Privacy | (√) |
| | C6. Anonymity | √- |

### 2.1.4. Standards Used

The SET protocol is based on several existing standards and algorithms. The following diagram illustrates how these are based on each other.

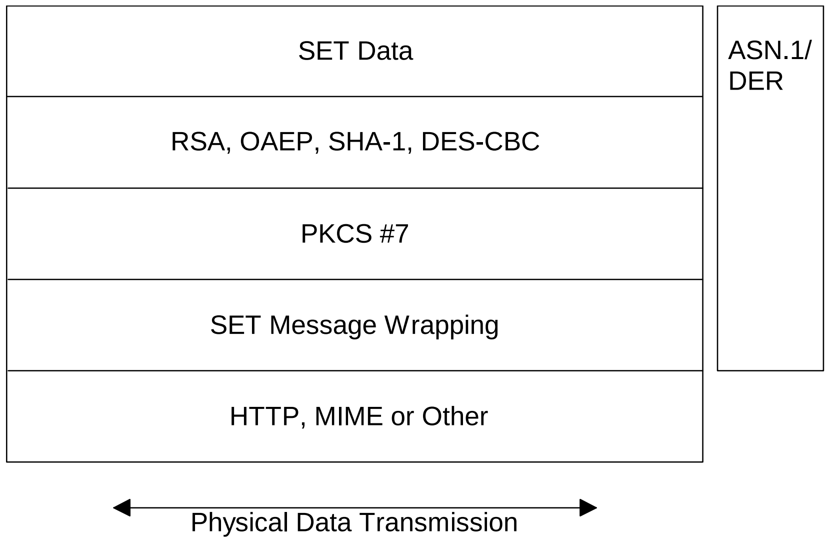


Fig. 2.2: Standards Used

The most essential cryptography standards used are the following. The RSA public-key algorithm is used for all public key operations and certificates. Highly sensitive data is additionally obfuscated with the Bellare-Rogaway Optimal Asymmetric Encryption Padding (OAEP) method [MaVi97b, p.88]. The standard algorithm for all hashes used is the Secure Hash Algorithm (SHA-1). For protection of sensitive financial data (for example the payment instructions) the standard encryption algorithm is the Cipher Block Chaining (CBC) mode of the Digital Encryption Standard (DES) [MaVi97b, p.65].

For representing cryptographic data and parameters the Public Key Cryptography Standard (PKCS #7) is used. This promotes the interoperability and ability to upgrade [MaVi97b, p.45]. For example if the message contains signatures, all certificates and CRLs needed by receiving party are included in the PKCS #7 envelope [MaVi97b, p.76].

In message wrapping [MaVi97b, p.47], information is included which is presented to the receiver of the message at the very start of message processing, for instance the message type and unique identifiers.

All messages are defined using the ISO/IEC and ITU-T Abstract Syntax Notation (ASN.1) standard and are encoded using the Distinguished Encoding Rules (DER) to permit unambiguous encoding [MaVi97b, p.45].

Depending on the transport mechanism, which is not defined in the SET specification, the message may be further wrapped, for instance with a MIME or HTTP header.

## 2.2. The Customer Entity

Whereas the previous section deals with the SET system in general, this section focuses on the customer entity. At first the flow of messages from the customer's point of view is described. Then the data structures of these messages are explained in more detail. The last paragraph lists essential problems of the customer entity. Removing these problems is then the subject of the following chapters.

### 2.2.1. Assumption

In the SET specification customer certificates that all other parties will accept are optional (X.509 certificates; see [MaVi97b, 207ff]). Because this is only a temporary option to make a first implementation of SET easier [MaVi97a, p.35], it is assumed hereafter that the customer is in possession of a certificate.

### 2.2.2. Flow of Messages

First the customer must register at the customer certificate authority (CCA) to get a certificate for his public signature key. This registration process is needed only once. It is not further described in this work. After registration, the customer can use the same certificate for every shopping process. The shopping process is executed every time the customer wants to order goods and pay for them with a credit card [MaVi97a, p.56]. All messages that are part of the SET protocol or belong to a pair of messages, the customer's request message, called Req, and the response message, called Res.

The shopping process begins with the browsing through the offer of a merchant, the selection of goods and the filling out of an order (see fig. 2.3). These activities belong to stages one to three in the diagram of paragraph 2.1.2. To make clear that they are not part of the SET protocol they are shown in gray whereas the SET messages are shown in black. The SET protocol requires that after these stages, both customer and merchant have the same bit stream of the filled out order (see paragraph 2.1.2).

The fundamental messages sent between the customer and the merchant are the purchase messages, PReq and PRes. The message PReq contains all data needed by the merchant and the acquirer to process the order. The message PRes informs the customer about the status of the order processing, e.g. whether the merchant has already received payment.

Both the message pairs PInit and Inq are optional. The purpose of the message PInitReq is to request certificates and certificate revocation lists (CRLs) for the customer. A necessary certificate is for instance Cert-PE, the certificate for the payment gateway's public encryption key. If this message pair is omitted, then it must be possible for the customer entity to obtain certificates through some other means, such as CD-ROM [MaVi97b, p.73]. The optional message InqReq requests information about the status of the order processing. This request is needed if the merchant sends the message PRes before the payment is finished. In this case, the customer does not get told when the payment will be finished and has to keep polling the merchant with the message InqReq. The response InqRes has the same structure as PRes.

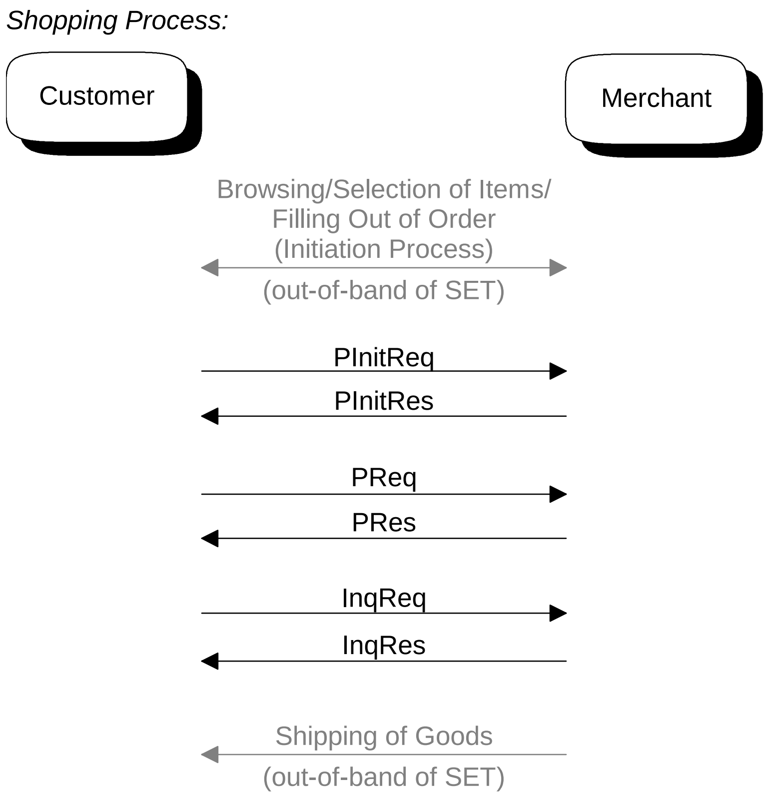


Fig. 2.3: Flow of Messages (Shopping Process)

In the following section the data structures of all these messages are described. Furthermore some messages sent between the merchant and the acquirer and used in the following chapters are described to make it easier to understand the previous messages.

### 2.2.3. Data Structure of Messages

The following diagrams illustrate the structure of the messages sent between the customer and the merchant. For simplicity, data not essential for a particular message are not illustrated, e.g. certificates or data for message wrapping. Whereas in the SET specification the data structures of the messages are described with tables [MaVi97b, p.263-434] here they are illustrated with structured trees. The symbol '⅄' indicates that the element shown above it is a data structure which contains the elements shown below it.

The data structure of the message PInitReq begins with the element RRPID, a request/response pair identifier that uniquely identifies a request to enable the relation of a response message to its request message. The elements LID-C and LID-M are local transaction identifiers used by the customer and the merchant, respectively. The element Chall-C is a challenge variable to ensure freshness of the response. Thumbs identify relevant certificates and CRLs already held by the customer to make the sending of them together with the response message unnecessary. With the elements Language, BrandID and BIN the customer defines his language, the chosen payment card brand and his bank identification number. The element PIRqExtensions defines optional extensions. The main purpose of the

PInitReq message is to request the certificate Cert-PE, which contains the payment gateway's public encryption key. This key is needed to construct the next request message PReq.

PInitReq

RRPID, Language, LID-C, [LID-M], Chall-C, BrandID, BIN, [Thumbs], [PIRqExtensions]

Fig. 2.4: PInitReq Data Structure

In the message PInitRes (fig. 2.6) all transaction identifiers are put together in the element TransIDs. XID is a unique global identifier. PReqDate is the date of the purchase request. If the element Thumbs of PInitReq indicates that the customer does not have the certificate Cert-PE, this certificate is included in the message wrapper of PInitRes. In the following diagrams operators are shown in bold. Here the operator 'S' indicates that the message is signed by the merchant.

PInitRes

**S**(M, PInitResData)

TransIDs, RRPID, Chall-C, Chall-M, [BrandCRLIdentifier], PEThumb, [Thumbs], [PIRsExtensions]

LID-C, [LID-M], XID, PReqDate, [PaySysID], Language

Fig. 2.5: PInitRes Data Structure

The following diagram (fig. 2.7) illustrates the message PReq. The structure of this message is the most complex of the SET protocol. It consists of the payment information (PI), the order information (OI) and a signature (PISignature), which is the signature of the PI and the OI and therefore called a dual signature.

The links between several elements of this message are more comprehensible if it is made clear which elements belong to the structure PI, which elements belong to the structure OI, and which elements belong to the signature. In the diagram all elements belonging to the signature are in a blue frame. The element DD(PIData) in the structure OI enables the merchant to verify the dual signature without knowledge of the elements of the PI. The element DD(OIData) in the structure PI enables the acquirer to verify the dual signature correspondingly.

The payment information (PI) is encrypted which is indicated by the operator 'EX'. The PI consists of two parts, the PIHead and the PANData. It is divided into two parts to enable stronger encryption to the more sensitive PANData. The encrypted PI is tunneled from the customer through the merchant to the acquirer. In the diagram, the part that describes the encryption is a red frame. The essential data of the PI is in a green frame.

The order information (OI) is sent to the merchant in unencrypted form. The essential data of the OI is in a yellow frame. At the bottom of the diagram the order description and corresponding data (HODInput) is in a gray frame. Not the data of the HODInput itself but only its hash is sent to the merchant and the acquirer.

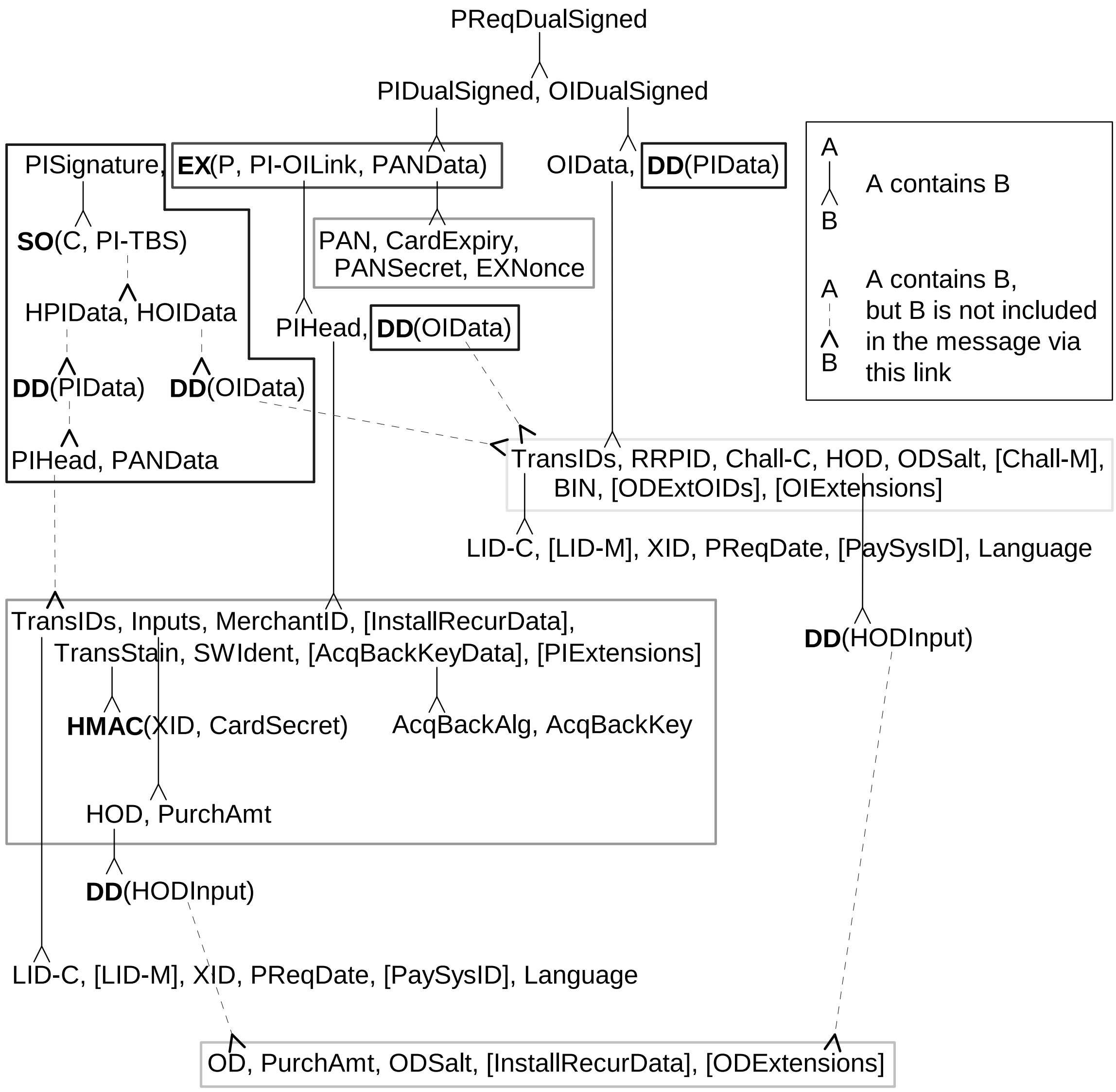


Fig. 2.6: PReq Data Structure

To make the linking within the message PReq more clear, a simplification is illustrated in the following diagram. For this the structure of the dual signature and all links corresponding to it are omitted, repeatedly encapsulated data structures are resolved and only the essential data of the PI and the OI is shown.

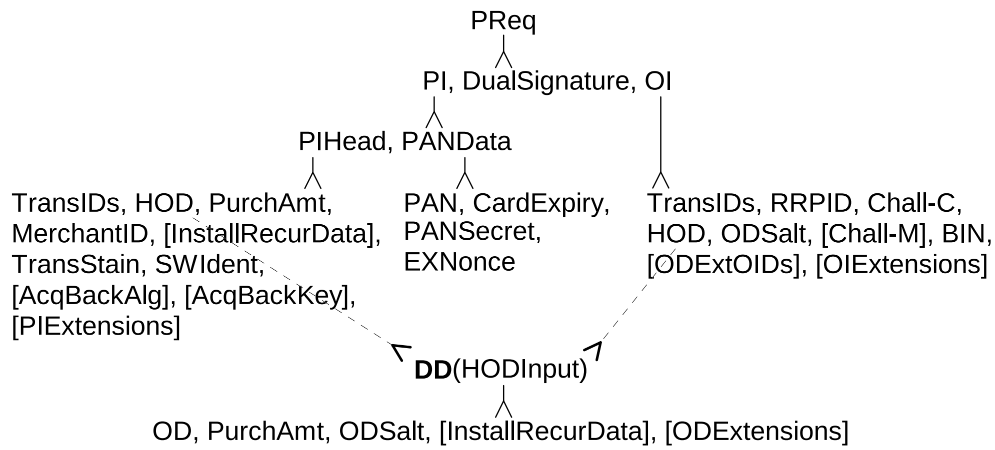


Fig. 2.7: PReq Data Structure (simplification)

The response message PRes (fig. 2.8) informs the customer about the status of the payment transaction processing. The essential data is the sequence PResPayload+. The '+' indicates that the structure may contain one or more instances, one per authorization performed.

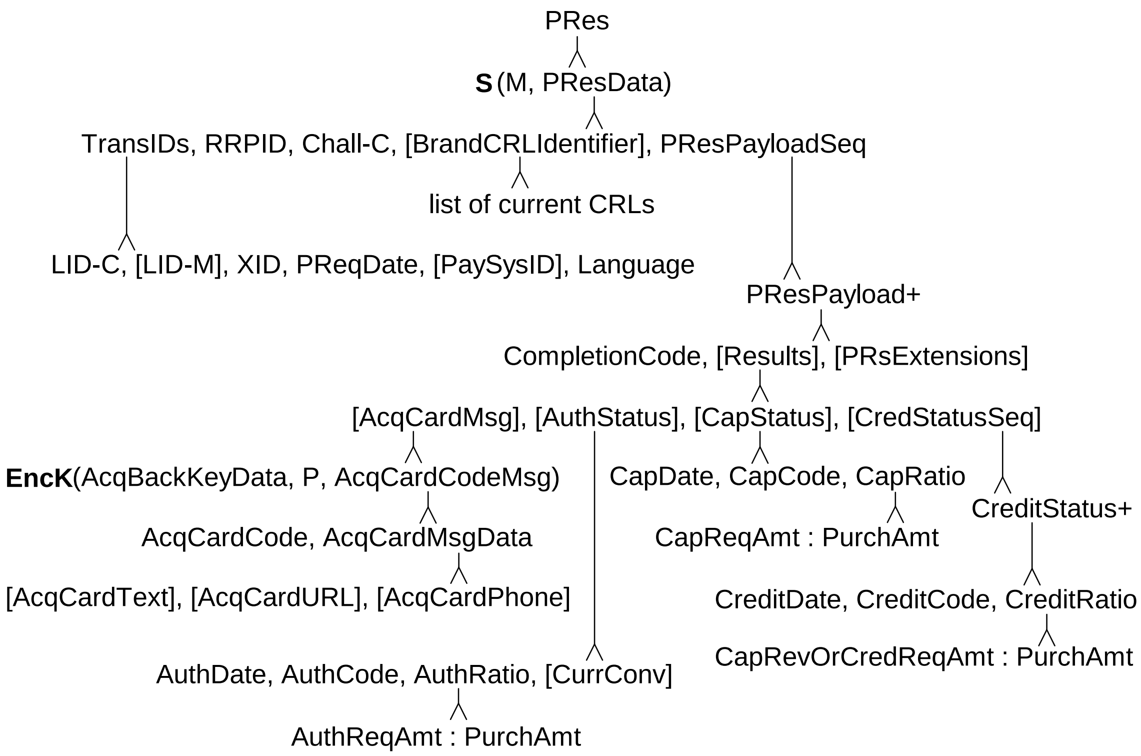


Fig. 2.8: PRes Data Structure

The element CompletionCode indicates whether the order was accepted and what the status of the processing is. The status of the processing is then given in more detail in the data structures AuthStatus, CapStatus and CreditStatusSeq.

After the customer has received a purchase response, he can optionally request the status of processing again with the signed message InqReq (fig. 2.9). The content of the response message InqRes is identical to the message PRes.

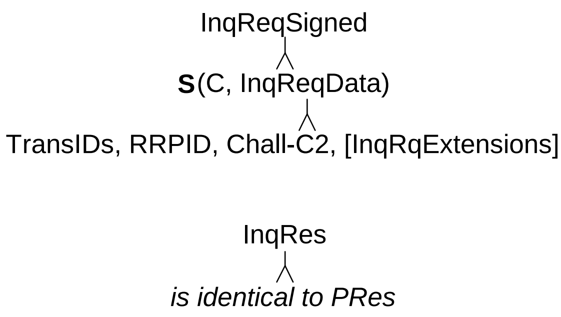


Fig. 2.9: InqReq and InqRes Data Structure

In the following the structure of the messages AuthReq and AuthRes sent between the merchant and the acquirer are illustrated.

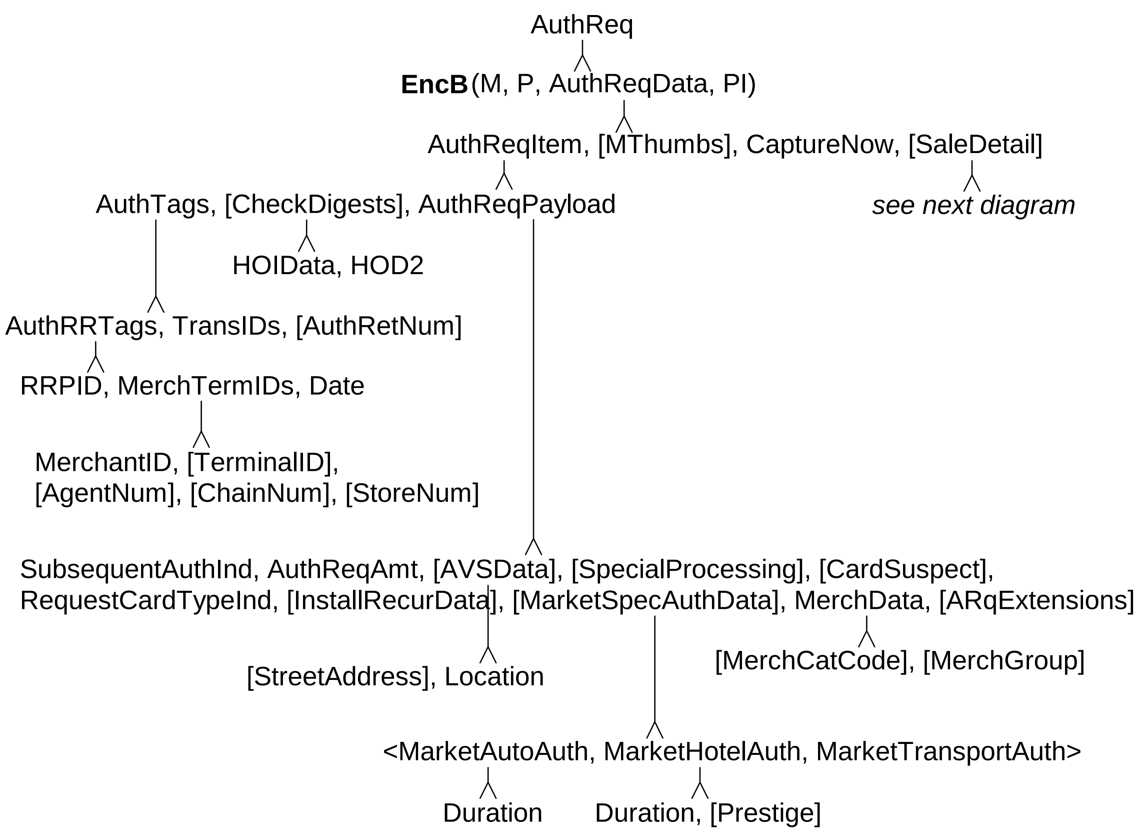


Fig. 2.10: AuthReq Data Structure

The message AuthReq contains a hash of the order information (HOIData) and a hash of the data HODInput (HOD2). Both hashes are computed by the merchant and then compared by the acquirer with the corresponding hashes in the structure PI to verify the agreement between the customer and the merchant. If the merchant wants to authorize and capture payment with a single request message, he sets the element CaptureNow to ’true’.

The data structure SaleDetail [MaVi97b, p.285] is an optional element of the message AuthReq. It contains very detailed information specific for certain purchases.

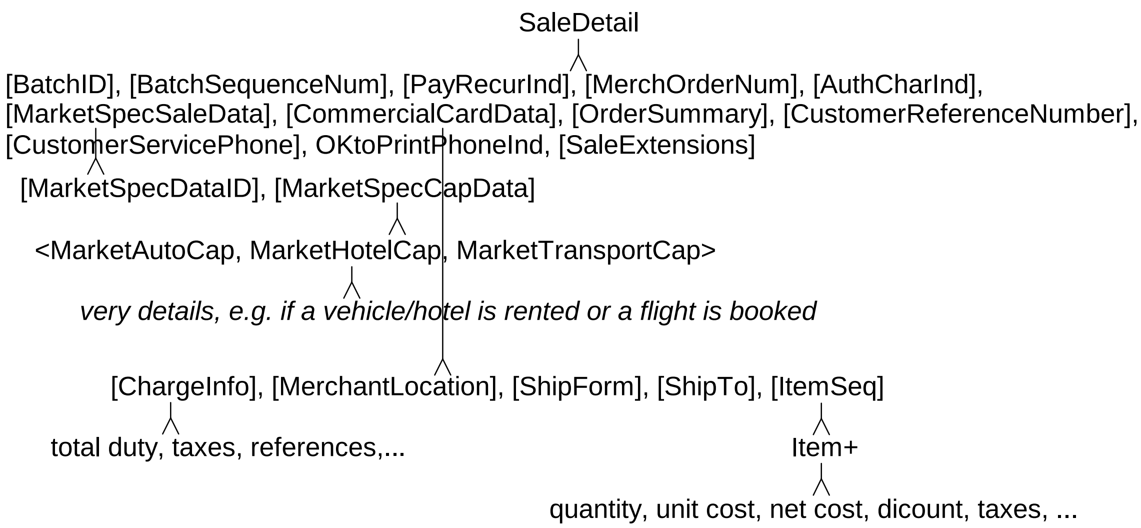


Fig. 2.11: SaleDetail Data Structure

On the basis of the message AuthRes shown below, the message PRes is subsequently constructed by the merchant. Essential data is the AuthAmt, which defines the authorized amount, and the AuthCode, which indicates the outcome of the authorization processing.

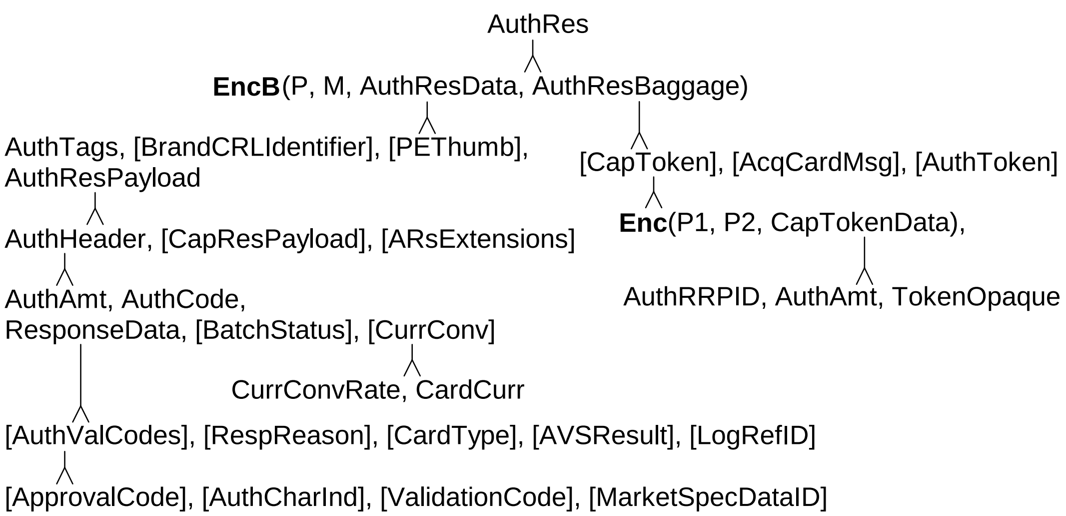


Fig. 2.12: AuthRes Data Structure

### 2.2.4. Problems

In the following, two problems of the customer entity are described. The first is the security of the customer entity, which is beyond the scope of the SET specification. The second is the mobility of the customer, a desirable property of the customer entity. It is not demanded by the SET specification but affects the design for an implementation just as much as the achievement of its security.

**a) Lack of Security**

SET provides secure transactions over open networks. That means that the communication is secure, even if the connection between the customer and the merchant is an insecure channel. But the specification of SET does not describe how the entities of the protocol have to be secured. "The following are outside the scope of this specification: [...] Security of data on cardholder, merchant, and payment gateway systems including protection from viruses, Trojan horse programs, and hackers" [MaVi97a, p.11].

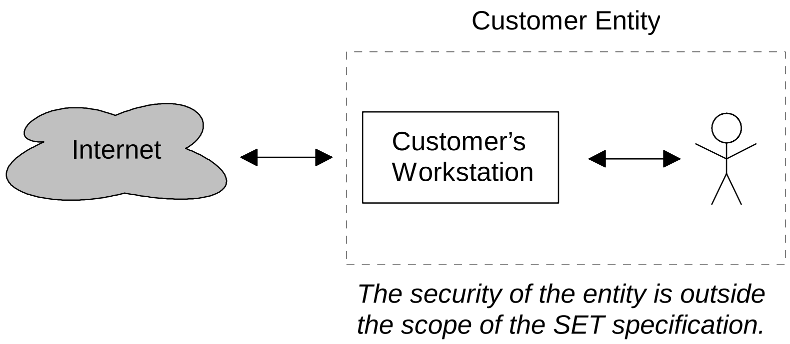


Fig. 2.13: Lack of Security

An implementation of the SET customer system on an ordinary customer's workstation without suitable means for its protection leads to a vulnerable system. Several types of attacks are possible in certain circumstances. Some types of attacks are shown in the following table.

| Type of Attack | Intrusion Point at Customer's Entity |
|---|---|
| Online Attack | Internet Connection |
| Trojan Horse | Installation of Software |
| Virus | |
| "Trojan Horse" in Hardware | Installation of Hardware |
| Unauthorized Login | Intranet Connection, User Interface |
| Hardware Tampering | Workstation Hardware |

If remote access to the workstation via the Internet is possible for authorized persons and the protection against unauthorized persons is not sufficient, online attacks are possible. Trojan horses and viruses can be brought into the workstation through installing of applications that are not sufficiently checked. The same is possible on the hardware level, if new hardware devices are added to the workstation. Analogues to online attacks, if the legitimation of users is not sufficiently checked, unauthorized logins may be possible via an Intranet connection or even via the user interface. Furthermore tampering with hardware is possible, if the physical access to the workstation is not sufficiently restricted.

Against several attacks suitable countermeasures exist, but at least against viruses and Trojan horses no today's countermeasure in software is sufficient for ordinary workstations. To protect a security critical application on a workstation without additional hardware, the operating system must be trusted to separate different applications such that untrusted applications have no access to the execution code, stored data, internal data flow, etc. of security critical applications. Ordinary operating systems are not regarded to be trusted for that.

The SET specification proposes to secure sensitive data through encryption, the use of firewalls and perhaps tamper-resistant hardware [MaVi97b, p.39]. Only encryption or the use of firewalls does not achieve complete security. The encryption leads to the problem that the encryption key also has to be secured. Firewalls can be broken through with Trojan horses (see above).

However, it is not the goal of this work to carefully analyze why an ordinary workstation cannot be trusted. In the approach of chapter four the assumption is made that the customer's workstation is untrusted. This is a given starting point for the main work.

Another assumption and starting point in chapter four is the existence of certain hardware devices that can be trusted. These trusted devices are described in the next chapter.

**b) Lack of Mobility**

If the SET customer implementation is a software-only solution on a local workstation, a disadvantage is the lack of mobility. The customer can use only his own workstation where his personal SET data, in particular the signature key and his account information, is stored. It would be desirable for the customer to be able to make SET payments at any workstation where the SET customer software is installed.

# 3. Trusted Devices

## 3.1. Overview

### 3.1.1. Definition of Trusted Device

The term *device* refers to a hardware extension that can be attached to a computer to extend its functionality. A device is called *trusted* if someone trusts the device to fulfill given requirements. A trusted device therefore must always be seen in connection with a given person who trusts and given requirements that are assumed to be fulfilled.

In this work, unless stated otherwise, the person who trusts is the customer and the requirements imposed on the trusted device are the following.

i. *Secret Data*
   The data on the device is kept secret to the extent that the cost for stealing data is expected to be higher than the profit the fraudulent person can make with it.

ii. *Correct Functionality*
   It is not possible to manipulate the functionality without being detected by the person or other devices it communicates with to the extent that the cost for manipulation is expected to be higher than the profit the fraudulent person can make with it.

### 3.1.2. Classification and Reasons for Trust

There are different ways of classifying trusted devices and different reasons why they can be trusted.

*Stationary or Mobile*
A device is either stationary or mobile. Mobile devices can be easily carried on one's person.

*Personalized*
A device can be personalized. This means that the device stores personal data of a user.

*Public or Private*
Devices are either public or private. A private device is used under the control of only one person. A public device is used by multiple persons.

*Tamper Resistance*
There are also tamper-resistant devices. Their hardware is designed in a way that makes them resistant to tampering.

Different types of trusted devices have different reasons why they can be trusted. Tamper-resistant devices can be trusted, for example, to keep data secret because of their hardware design. Private devices can be trusted, for example, to work correctly (e.g. a private calculator) by the person who always carries the device because he knows that the device was not in anyone else's hand. But such a private device can not be trusted to keep data secret because the owner does not know whether it will be robbed one day.

In addition to different types, there are different purposes of trusted devices. There are data storage devices, processing devices, user input devices and output devices. A device can also fulfill a combination of these purposes.

### 3.1.3. Examples

There are many different examples of trusted devices. Chip cards, especially smartcards, are regarded to be suitable to support Internet payments because they are mobile, cheap, and already used for usual payments. Therefore they are described in an extra chapter.

The use of chip cards requires card reader devices. These devices can be trusted, if they are tamper-resistent or private. Most today's readers used are stationary and not all have a display and a keypad (e.g cheap card readers used for workstations). If they do not have a display and a keypad or are not trusted, additional trusted user input/output devices for the smartcard may be needed (cf. section 3.3). An existing example is a mobile and private smartcard display device (the Visa Viewer, [Hend97, p.174]).

More functional devices than smartcards do not only store and process data, but also include a display or a keypad. Examples are trusted versions of mobile user devices, e.g. mobile phones, PDAs or multifunctional watches.

## 3.2. Smartcards

A smartcard is a plastic card the size of a conventional credit card. Besides smartcards there are also other types of plastic cards. The exact format of a plastic card is specified in ISO 7810. Examples of different types of these cards are embossed cards, magnetic stripe cards, optical cards and chip cards. Chip cards can be divided into memory cards, memory cards with access control and cards with a microprocessor. The latter is also called a smartcard. An overview of cards can be found in [ZoOt94], an introduction to chip cards in [Meye97], [ScKo96], [VoVo96] and [ZoOt94].

### 3.2.1. Card Types

*Embossed Cards*
Embossed cards are simple plastic cards with embossed characters and numbers. The embossed data on the card can be copied by pressing the card and a piece of paper together. The data is then embossed into the paper.

*Magnetic Stripe Cards*
Magnetic stripe cards are cards where several hundred bytes of data can be stored on an embedded magnetic stripe on the back of the card. With these cards, read and write access is possible.

*Optical Cards*
Optical cards, also called laser cards, can store several megabytes of data. They consist of two layers, a highly reflective layer and a non-reflective layer behind each other. To write, laser diodes melt little holes into the top layer. The sequence of reflective and non-reflective areas represents a stream of bits. Optical cards can be written only once.

*Memory Chip Cards*
Memory chip cards are very similar to smartcards but do not have a microprocessor. Smartcards are described in the next paragraph. Here, a memory chip card does not refer to the PCMCIA card used for memory extensions, but rather a card the size of a credit card. The memory chip cards can only store data but are not able to process it. Some memory cards have an included access control. This access control, e.g. a PIN check, is implemented with hardware logic, a wired logic without a processor. An example of this type of chip card is the prepaid card for public phones. Sometimes these cards are also called smartcards, but far better access control can be achieved with chip cards that contain a microprocessor. The following figure shows the basic components of a memory chip card with access control [ScKo96, p.23].

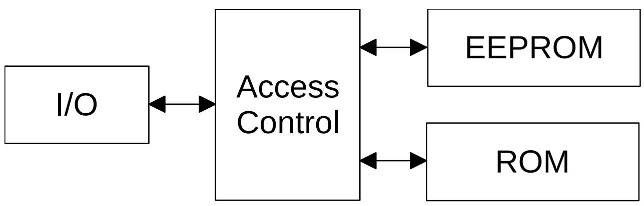


Fig. 3.1: Basic Components of a Memory Chip Card

### 3.2.2. Smartcard Properties

The term smartcard has no general exact definition. In this work a smartcard refers to a card having the characteristics described below. Not all properties described here are exclusive properties of smartcards. Some are properties of chip cards or plastic cards in general.

#### Mechanical Properties

A smartcard is 0,76 mm thick and its size is the specified ID-1 format, which is the size of a conventional credit card. Because this size is too large for some newer applications, e.g. cards for mobile phones, smaller formats ID-00 and ID-000 were defined. Smartcards must fulfill similar mechanical requirements as plastic cards without a chip, for instance bending properties. The specifications for contact chip cards are defined in ISO/IEC 7816.

#### Chip Components

A smartcard can store not only data and communicate with a card reader such as a memory chip card, it can also process data. Therefore its chip also has a CPU and RAM. It is a one-chip computer. The following figure shows the basic components of a smartcard chip [ScKo96, p.12].

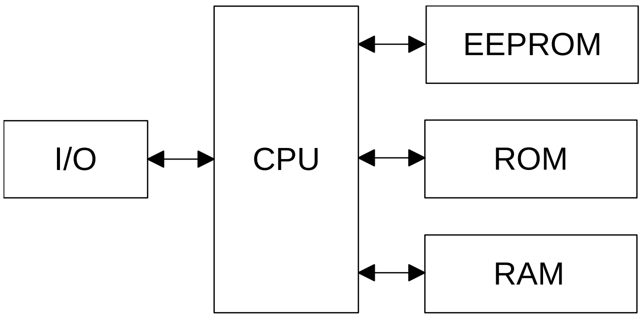


Fig. 3.2: Basic Components of a Smartcard Chip

*Memory*

The smartcard chip has three different kinds of memory: ROM, EEPROM, and RAM. The ROM contains the card operating system and constant data. Its current capacities are from a couple of KByte up to 20 KByte. The non-volatile EEPROM is used to store received or generated data that should remain if the card is not supplied with power. The average storage durability is 10 years or one million write/delete operations. Current capacities are between 256 Byte and 64 KByte. The write access of the RAM is much faster than that of the EEPROM. Current capacities are from 64 to 796 Byte [Schw97, p.3,4].

*CPU*

Current CPUs of smartcard chips are based on the conventional processors 80C51, 6505 or H8. Some also have a coprocessor to accelerate specific tasks, in particular cryptographic functions.

*I/O Data Transmission*

Regarding the transmission of data between the card and the reader, smartcards can be divided into two groups, the contact smartcards and the contactless smartcards. The contact smartcards have metal contacts on their surface and have to be inserted into a smartcard reader. The contactless smartcards have no contacts but an embedded coil, which is inductively excited by an external electromagnetic field [ZoOt94, p.51]. They require only close proximity to an antenna.

*Chip Example*

An example of a specialized chip is the Siemens SLE44C200. It has a 8051-based CPU and a coprocessor to speed up RSA encryption. The memory capacity is 10 KByte ROM, 2.5 KByte EEPROM and 256 plus 350 Byte RAM for processor and coprocessor, respectively.

**Security Properties**

The smartcard is a mobile device that can be trusted by its owner, if it is sufficiently protected. Here, the term sufficiently means that the cost of attacks is expected to be higher than the profit the fraudulent person can make with it (cf. 3.1.1). For example, if the smartcard is used for payments, beside its protection a limit for possible payments is prerequisite such that the customer can trust his smartcard. Several means for smartcard protection are given in the following.

A card chip must work properly only within specified conditions, for instance with a 5 V power supply and a 5 MHz clock rate, but it must be safe under any conditions. Possible fraudulent attacks and their countermeasures can be found in [VoVo96, p.106f] and [ZoOt94, p.46]. The following are some measures and properties that make a smartcard secure.

*Protection against Counterfeiting*

The smartcard reader can reduce the risk of accepting a fake card by checking response times of specific functions of the smartcard.

Furthermore the authenticity of a smartcard can be checked with the challenge response method. For this the card must have stored a secret key whose existence can be checked by verifying the response of the smartcard.

*Protection against Software Attacks*

Stealing and altering of data is prevented by the complete control over the input and output. The input and output can either be controlled by the card operating system or by the application itself, if there is no operating system on the card. In contrast to ordinary workstations the operating systems of smartcards can be trusted because they are less complex and security is an important aspect of their design. Furthermore all applications on the card can be trusted by the customer because no installation of (untrusted) applications is possible. Hence with (today's) smartcards viruses and Trojan horses are no danger (cf. 2.2.4).

Furthermore unauthorized access and hence the disclosure of data is avoided in many cases by restricting access with a PIN and by checking the reader/computer with the challenge response method.

*Protection against Hardware Attacks*

Many measures exist for smartcard chips to prevent a hardware attack. For example, metal layers used as a protection shield prevent micro-probing. Removing these layers damages the chip if the layers are part of the chip components. Furthermore, unlike in normal chips, a disorder of the chip design makes an analysis more difficult.

Moreover a disclosure of data is possible without intrusion into the chip. The tracking of instruction sequences by measuring the power consumption can be avoided by generating erratic consumption and using dummy operations.

### Application Examples

Because smartcards can be trusted as described above and because of their mobility they can be useful for applications of many areas, for instance electronic payments, electronic tickets, identification, data collection or combinations thereof.

Currently, specialized smartcards are produced for each application. In the future multifunctional cards could reduce the number of cards a single person has to carry.

## 3.3. Important Further Requirements

As described in paragraph 3.1.1. a device is called *trusted* if someone trusts the device to fulfill certain requirements. These requirements are that it keep data secret (i.) and that the functionality is correct (ii.). For certain applications additional requirements on trusted devices are important.

iii. *Trusted User Input/ Trusted User Control*
For some applications the device needs input from the user. The input may be some text, a number (e.g. a PIN, see requirement vi.) or simply a confirmation "yes" given by pressing a single button (user controls device). The latter is useful for example before a device generates a digital signature. A facility for user input could be a keypad or a single key.

iv. *Trusted Output to the User*
Some applications require that the user of the device be able to check certain data. For example, if the device generates a digital signature, the user should be able to check the data to be signed. A facility could be a little screen or in some cases perhaps a loudspeaker.

v. *Device Identification*
For some applications the user must be able to ascertain whether the device is really the trusted device and not a fake one. This is important, if the device is used for crucial operations or if the user inputs sensitive data.

vi. *User Identification*
Some applications require that the device identify its user. For example, if the device asks its user for confirmation before it generates a digital signature (see requirement iii.), the device should ensure that the user is the legitimate user. A simple measure for user identification is a personal identification number (PIN). Additional security can be achieved with biometrics, but they require specialized and expensive hardware [PPSW96, p.9].

The requirements above can be achieved with facilities attached directly to the trusted devices or with additional devices. Obviously these facilities or additional devices as well as any additional connections must also be trusted.

In the following chapter trusted devices are used to solve the problems described in paragraph 2.2.4.

# 4. Trusted Device-Support for Customer Entity

## 4.1. Approach

### 4.1.1. Approach Used

The goal of chapter four is to solve the problems mentioned in paragraph 2.2.4. The main problem is the lack of security. The security problem occurs, because the workstation on which the SET customer software is implemented can not be trusted by the customer. This in one of the main assumptions in this approach for the further work.

| Assumption: The workstation is untrusted. |
|---|

It is given as part of the starting point for this work that the security problem shall be solved with an additional trusted device. The second problem mentioned in paragraph 2.2.4 is the lack of mobility. Because this problem can be solved with the use of mobile devices, it is also given that the trusted devices to be used shall be mobile. An existing example for a trusted mobile device is the smartcard (for description of smartcard see section 3.2). Since a fast further development of mobile devices is expected, the work should not be restricted to smartcards but to trusted mobile devices in general. The requirements on a device needed for its trustworthiness are described in chapter three. The trust in the device is one of the main assumptions for the further work. In the following this additional device is called trusted device.

| Assumption: The additional (mobile) device is trusted. |
|---|

Current mobile devices have too little resources for an implementation of the entire SET customer system. This is an assumption for the further work.

| Assumption: The entire SET customer system does not fit on the trusted device. |
|---|

Because the entire system does not fit on the trusted device, it has to be discussed whether and how a split of the system is possible. Therefore the work focuses on the following problems. These problems are discussed each in a separate chapter.

For a split the determination of parts to be implemented on the workstation and parts to be implemented on the trusted device is needed. As mentioned above the main goal of this work is to solve security problems. Different parts can be determined which are independent regarding the fulfillment of security purposes. With this an implementation of these parts on the trusted device could guarantee the corresponding security properties. Such different parts are determined in the following section 4.2.

If not all determined parts can be implemented on the trusted device, it must be discussed which are the most security critical parts of the SET customer system. Therefore possible attacks are determined in the section 4.3. But only the determination of attacks would not be sufficient for an approach to define what must be done by the trusted device. Attacks can draw attention to security critical parts. It is never clear if all theoretically possible attacks are determined. Therefore only with the prevention of determined attacks security properties can not be guaranteed.

For several critical parts methods for implementation are developed in the section 4.4. For particular trusted devices possibly the entire part cannot be implemented. For this case the security properties of the part cannot be guaranteed completely, but several methods are developed that perform additional tasks to achieve at least some benefit. For each method the benefit and the requirements on the trusted device are given. As an example for the combination of different methods in section 4.5 implementation proposals are given for two scenarios.

### 4.1.2. Previous Approach (Abandoned)

The idea of this approach is that different hardware properties regarding security exist. These properties guarantee different security properties for the software. For example the guarantee of secrecy and guarantee of integrity are different hardware properties. Furthermore different elements and procedures of the SET customer system require different security properties of the hardware. Some elements must be secret and for others only integrity is required.

In this approach the security requirements of single data elements and procedures were analyzed. The goal was to determine with this which element must be stored and which procedure must be executed on which device.

The approach were abandoned, because the achieved results were not usable for a split of the software system. For example, if a single element requires secrecy, it is not sufficient only to store this element on a device which guarantees secret storage. The secrecy gets lost by further processing of it in devices which do not guarantee secrecy. The results of this approach give no information about dependencies between different elements and procedures and no information about different parts which have to be secured together. Therefore the new approach is made which is described in the last paragraph.

## 4.2. Determination of Independent Parts

### 4.2.1. Split of System Into Independent Parts

Different security methods are used by the SET customer system for several messages. In summary the essential security methods used are the signature generation and verification, the encryption and the verification of certificates.

The signature generation is used for the generation of the messages PReq and InqReq. In the message PReq the structures PI and OI are signed (see fig. 2.6), in the message InqReq all elements are signed (see fig. 2.9). The encryption is used only for the generation of the message PReq (operator EX in fig. 2.6). It is a hybrid encryption. Very sensitive data of the structure PI and the symmetric key are encrypted with the public encryption key, the rest of PI is encrypted with the symmetric key. Decryption is used in the message PRes (operator EncK in fig. 2.8). But since the data to be encrypted is an optional element and furthermore not regarded to be very essential for the payment, it is not observed in the following. The signature verification is used to verify the merchant's signature in the messages PInitRes and InqRes. Furthermore certificate verification is needed to verify the merchant's public signature key, to verify the MerchantID and to verify the acquirer's public encryption key. The last two elements are needed to generate the message PReq.

Between several security methods an interdependence exist. The security achieved by the encryption is dependent on the certificate verification of the acquirer's public encryption key. The signature verification is dependent on the certificate verification of the merchant's public signature key. The signature generation is not dependent of another method. With this independent parts of the SET customer system regarding security are the signature generation, the encryption plus the certificate verification and the signature verification plus the certificate verification. These different parts are illustrated in the following figure 4.1.

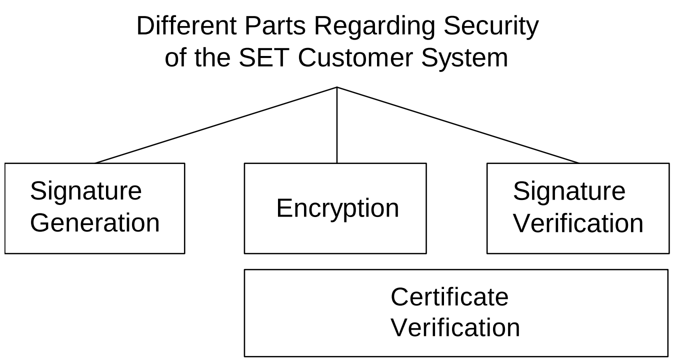


Fig. 4.1: Dependency of Parts

The different parts are candidates for an implementation on a trusted device. If a part is implemented completely on the trusted device, the serving of its purpose regarding security can be guaranteed. With purposes regarding security the fulfillment of security requirements of the customer is meant. These requirements of the customer are described in paragraph 2.1.3.

In the following table the purposes of the different parts are listed.

| |
|---|
| *Purpose of Signature Generation*<br>• only the customer can authorize a payment<br>(cf. requirement C1 in paragraph 2.1.3) |
| *Purpose of Encryption together with Certificate Verification*<br>• confidentiality of payment information<br>(cf. requirement C5 in paragraph 2.1.3) |
| *Purpose of Signature Verification together with Certificate Verification*<br>• customer can trust the response messages<br>(cf. requirement C3 in paragraph 2.1.3) |

The requirement C6 in paragraph 2.1.3 is not related to a particular part, because it is not fulfilled by processing but with a pseudo-identity in the customer's certificate

The requirement C1 is regarded to be one of the most important customer's requirements. Therefore the signature generation can be regarded as one of the most important parts to secure. Furthermore this part is not based on certificate verification. Because of this and the results of the following section, in section 4.4 at first methods are developed for the signature generation part.

### 4.2.2. Description of Part: Signature Generation

The generation of the signature in the message PReq consists of the following steps.

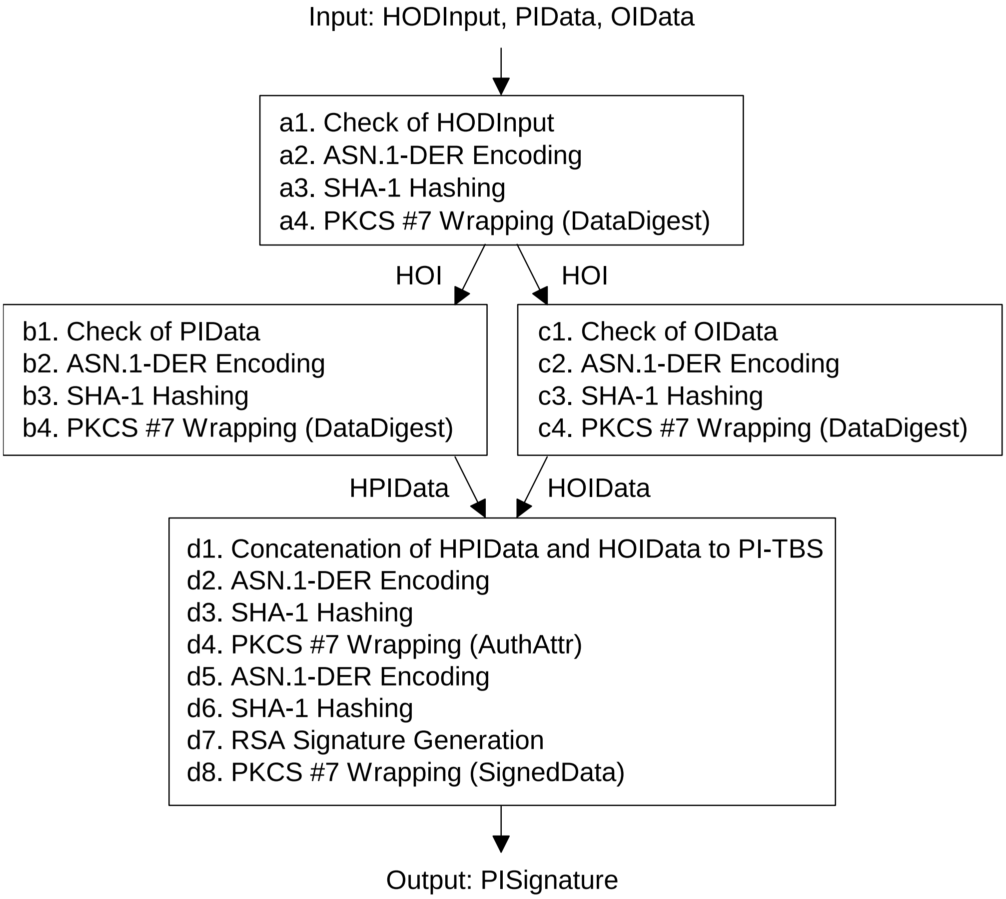


Fig. 4.2: Procedure for Generation of PISignature

Information on the ASN.1-DER encoding can be found in [MaVi97b, p.46]. Properties of the SHA-1 hash algorithm are described in [MaVi97, p.521]. The PKCS #7 formats are described in [MaVi97b, p.51-55] (in figure 4.2 and in the following, the abbreviation 'AuthAttr' is used instead of 'AuthenticatedAttributes'). The key size for the RSA algorithm is given in [MaVi97b, p.64] and the algorithm is described in [Schn94, p.281ff.].

In summary the following functionality is needed.

| *Needed Functionality* |
|---|
| • Check of Data<br>• ASN.1-DER Encoding<br>• SHA-1 Hash Algorithm<br>• PKCS #7 Wrapping (DataDigest, AuthAttr, SignedData)<br>• RSA Signature Algorithm |

"Check of Data" means that either the data is checked personally or that the corresponding fields are populated by a device and that this population can be trusted by the customer.

Furthermore the following data, which is not input data, is needed.

| *Needed Data* |
|---|
| • Private Signature Key<br>• PKCS #7 Headers for HOI, HPIData and HOIData<br>• PKCS #7 Header for AuthAttr<br>• PKCS #7 Frame for PISignature |

As long as the private signature key is not changed, the PKCS #7 headers and the header for AuthAttr are also static data. This means that the same data can be used for the generation of different PISignature's. The PKCS #7 Frame for the PISignature is static except for the included certificates and CRL's.

## 4.3. Attacks

As described in paragraph 2.2.4 several attacks against the customer entity are possible, if the customer's SET software is a software-only implementation on an ordinary workstation. Because in the approach (see 4.1.1) the assumption is made that the customer's workstation is untrusted here not the first step of an attack, the intrusion into the workstation, shall be analyzed but what are subsequent steps after the intrusion needed to attack the SET system.

The attacks described shall draw attention to security critical parts. Therefore not all theoretical possible attacks are determined, for instance, attacks that merely abort the communication but cause no fake payment are not discussed.

Because attacks with different intentions do not consist of completely different steps but may have common parts, the attacks are divided into prerequisite attacks and main attacks. After certain prerequisite attacks, different main attacks are possible. Main attacks result in fake payment.

Prerequisite attacks are described in paragraph 4.3.1 and denoted P1, P2, etc. Main attacks are denoted A1, A2, etc. and described in paragraph 4.3.2. An overview of all prerequisite and main attacks and their dependencies is given in paragraph 4.3.3.

### 4.3.1. Prerequisite Attacks

Prerequisite attacks are the common steps needed for several main attacks. The attacks were originally determined the other way round: first the main attacks were determined and then the prerequisite attacks. For easier reading the prerequisite attacks are described first. Because of this they were not determined separately but their need is a result of the determination of the main attacks.

Attack P1 is the prerequisite attack for all other attacks, the intrusion into the workstation. Attacks P2, P3 and P5 are attacking the signature generation part of the SET system (cf. 4.2). They are stealing needed data for main attacks. Attacks P6, P7 and P8 are manipulating the verification part of the SET system. Hence they enable the attacking of the encryption part with attack P4 or the attacking of the signature verification part (attack A8).

#### P1. Intrusion into Customer Entity

*Attack Description:*

If the SET customer software is installed on an ordinary workstation with an Internet connection, the intrusion of malicious programs cannot be ruled out as described in 2.2.4. These programs could then manipulate data or program code. Furthermore it cannot be ruled out that these programs might steal data by sending it back to the attacker.

Means for intrusion into the customer entity could be for example Trojan horses, viruses, online attacks, unauthorized local access as well as hardware attacks (stealing and changing hardware components).

*Possible Purpose:*

This attack is a prerequisite for all other attacks.

### P2. Stealing of Private Signature Key

*Prerequisite Attack:* P1

*Attack Description:*

The program code of the SET customer software is not a secret code. An attacker can purchase the SET customer software and analyze it. As a result, it cannot be ruled out that an intruded program (attack P1) would be able to find the private signature key in the customer entity and send it back to the attacker. Even if the key is not stored in clear text but in encrypted form, an attack of this kind is not precluded. It simply makes this attack more difficult and changes the problem to that of finding the encryption key.

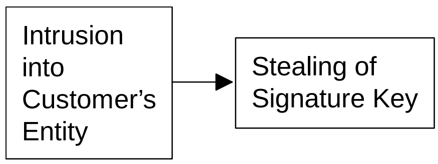


Fig. 4.3: Stealing of Private Signature Key

*Possible Purpose:*

This is the prerequisite attack for attack A1.

### P3. Stealing of CardSecret

*Prerequisite Attack:* P1

*Attack Description:*

The data CardSecret is secret data needed to generate SET messages (see [MaVi97b, p.56]), but it is not present in them. As the private signature key it is only present in the customer entity; attack P1 is necessary to steal the key. This attack is analogous to attack P2.

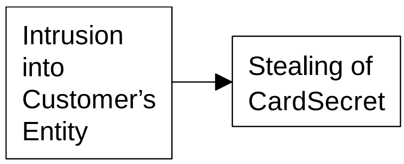


Fig. 4.4: Stealing of CardSecret

*Possible Purpose:*

This attack is prerequisite for attacks A1 and A2.

**P4. Stealing of Account Data**

*Prerequisite Attack:* P1 or P6 or P7 or P8

*Attack Description:*

The account data consists of the following elements:
- Private Account Number (PAN);
  including the Bank Identification Number (BIN)
- Date of Card Expiry (CardExpiry)
- Secret of Private Account Number (PANSecret)

There are two implementation alternatives for the customer software given in the SET specifications to make these data available. Either the PAN, CardExpiry and PANSecret are stored in a device or only the PANSecret is stored and the private account number and expiry date have to be entered in some form by the customer for every purchase.

In contrast to the private signature key or CardSecret the account data is also present in the message PReq. Therefore it can not only be stolen out of the customer entity but also by means of interception and decryption of a PReq message.

If the attacker intends to steal the account data in the customer entity and all data is stored in a device, this attack is analogous to attack P2. If the PAN and CardExpiry is always entered by the customer, the intruded program has to tap the customer input or it has to steal the data out of the field PANData shortly before it is encrypted.

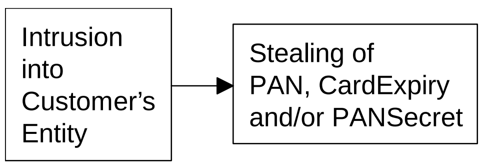


Fig. 4.5: Stealing of Account Data (a)

If the attacker intends to steal data by intercepting of the PReq message outside of the customer entity, one of the attacks P6, P7 or P8 is prerequisite. The attacker is then in the possession of the private equivalent of the public encryption key with which the customer encrypted his account data. After interception of the message PReq the attacker can decrypt the field PANData and steal the account data. The attacker can then encrypt the message with the acquirer's public encryption key and forward it to the merchant so that the disclosure of the account data will not be recognized.

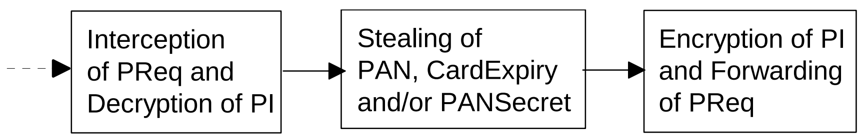


Fig. 4.6: Stealing of Account Data (b)

*Possible Purpose:*

This attack is prerequisite for attacks A1 and A2. It can also be used to facilitate attack A3.

### P5. Stealing of Signature Certificate and BrandID

*Prerequisite Attack:* possibly P1

*Attack Description:*

The certificate for the customer's public signature key and the BrandID are personal data but not secret, indeed the certificate is public data. Nevertheless the attacker must obtain this data somehow, if he wants to impersonate the customer. One possibility to obtain this data is to steal it directly from the customer entity (attack P1 is then necessary).

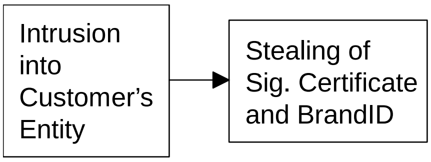


Fig. 4.7: Stealing of Signature Certificate and BrandID (a)

Another way to obtain the certificate is to tap any message that is signed by the customer, e.g. PReq or PInqReq. If the addressee does not have the certificate for the signature with which the message is signed, the certificate is attached to that message by the customer. The attacker must simply copy the certificate. If the message is encrypted by means which are outside of the SET specifications, for instance in a lower communication layer (e.g. a HTTP layer, see diagram in paragraph 2.1.4), it makes this attack more difficult. Interception and Decryption as described in P4 could then be a way to obtain this data. But as the data itself is not secret there are further possibilities. For example, if public available libraries for the certificates exist, which is not unusual when signatures are used, the attacker just has to request the certificate there.

Another way to obtain the BrandID is to tap the message PInitReq or PReq. In these messages the BrandID is a not decrypted element. Problems could be that these messages are encrypted outside of SET (the same is then valid as above) and that the used BrandID need not to be always the same since the customer may has the possibility to choose between several brands. But the attacker can also just try out several BrandID's since the total number of different payment card brands is very limited.

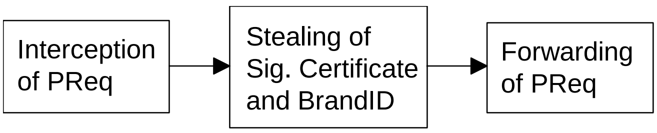


Fig. 4.8: Stealing of Signature Certificate and BrandID (b)

*Possible Purpose:*

This attack is prerequisite for attacks A1 and A2.

### P6. Manipulation of Verification Code and Changing of Not Verified Data

*Prerequisite Attack:* P1

*Attack Description:*

In the customer entity several certified data have to be verified. Specifically , these are the merchant's public signature key and the acquirer's encryption key. If an attacker wants the customer to use a fake key, there are three different ways to achieve that. These different ways are described as the attacks P6, P7 and P8.

The verification of certified data in the customer entity is a procedure where the signature of the corresponding certificate has to be checked. The result of a verification is either "the certificate is correct" or "the certificate is not correct". If the certificate is correct, the verified data can be used for further processing.

After the attack P1 the verification program code can be manipulated so that with a particularly given certificate the result is always "certificate is correct" independently of the certified content. Then the content of the particular certificate, which is non-verified data, can simply be changed into the desired data of the attacker.

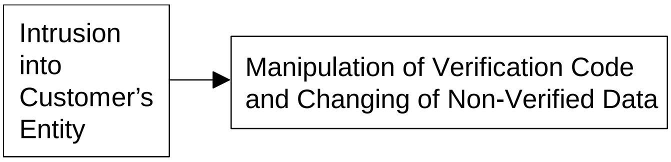


Fig. 4.9: Manipulation of Verification Code and Changing of Non-Verified Data

*Possible Purpose:* This attack facilitates attacks P4 and A8.

### P7. Changing of Verified Data

*Prerequisite Attack:* P1

*Attack Description:*

As described in P6 the verification of certified data is a procedure that determines whether the certificate is correct. In the following the verified data which has a correct certificate is stored in the customer entity for further processing. If after attack P1 this data, which has been verified, is changed, no manipulation of the verification code is needed. The difficulty is that the time to make this attack is much shorter than to make attack P6.

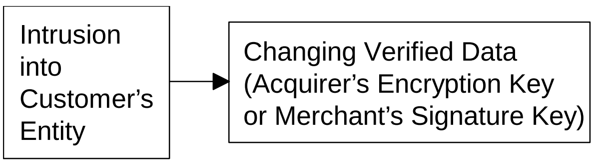


Fig. 4.10: Changing of Verified Data

*Possible Purpose:* This attack facilitates attacks P4 and A8.

### P8. Changing of Key of Trust Chain And Sending of Fake Certificates

*Prerequisite Attack:* P1

*Attack Description:*

A certificate is only correct if all certificates of its trust chain up to the root key are correct. But the verification procedure does not always check all certificates of the complete trust chain. As described in P7 the verification procedure stores verified data for further processing if the certificate is correct. This is also valid for the verified keys of a trust chain. Therefore the verification procedure checks only new certificates of a trust chain up to a certificate where its key has been verified.

After attack P1, the verification code (P6) is not manipulated nor the verified data itself (P7) but a verified key of its trust chain. This key can be any signature key of the trust chain, for instance the key of the geopolitical or brand certificate authority (GCA or BCA signature key), and also the root key, which is the key of the root certificate authority (RCA signature key).

After changing a verified key of the trust chain the attacker can send a certificate revocation list to the customer. Then he can send fake certificates, a certificate signed with the previously changed key and further certificates down to a SET end-entity certificate. This can then contain for instance a fake merchant’s public signature key or a fake acquirer’s public encryption key.

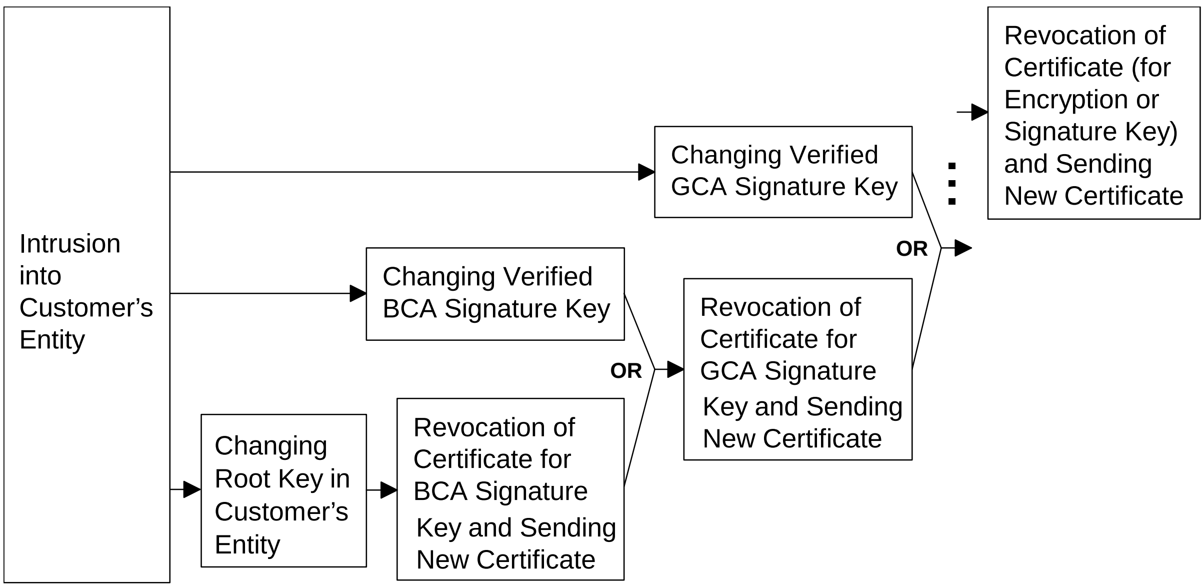


Fig. 4.11: Changing of Key of Trust Chain and Sending of Fake Certificates

*Possible Purpose:*

This attack facilitates attacks P4 and A8.

### 4.3.2. Main Attacks

All main attacks are resulting in fake payment (except for the last attack A8 that can be used in combination with other main attacks). With the attacks A1 and A3 the completely independent generation of fake payments is possible. With attack A2 all data needed for independent payments is stolen except for the data that is regarded to be at most protected against stealing. Therefore fake payment with this attack are still dependent from the customer's entity. With attack A4 the customer entity is used to generate a complete fake payment. The attacks A5, A6 and A7 do not generate additional fake payments but they are changing the content of payments intended by the customer.

#### A1. SET Customer Impersonation

*Prerequisite Attacks:* P2 and P3 and P4 and P5

*Attack Description:*

The SET customer entity consists on the one hand of well-known algorithms or obtainable software and on the other hand of secret or personal data specific for one customer. The identity of a customer within the SET system and therefore the legitimation for him to make SET payments lies only in the secrecy of his personal data. This personal data consists of the following data elements.

| *Personal Data* |
|---|
| *Cryptographic Data for Identification*<br>• Private Signature Key<br><br>*Account Data*<br>• Private Account Number (PAN);<br>including the Bank Identification Number (BIN)<br>• Date of Card Expiry (CardExpiry)<br>• Secret of Private Account Number (PANSecret)<br><br>*Further Personal Secret Data*<br>• CardSecret<br>(for Transaction Validation by non-SET systems, see [MaVi97b, p.56])<br><br>*Further Personal But Not Secret Data*<br>• Signature Key Certificate<br>• Identification of Payment Card Brand (BrandID) |

If all personal data of a customer is stolen, a complete impersonation of the customer within the SET system is possible.

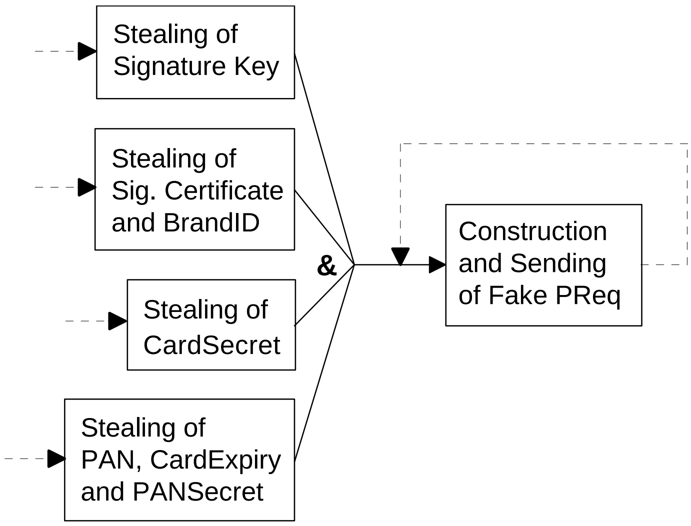


Fig. 4.12: SET Customer Impersonation

*Possible Intention of Attack:*

With this attack, goods or services electronically purchasable via the SET system can be obtained at the expense of this customer. Because the giving of a delivery address could easily expose the identity of the fraudulent person, in practice this attack would be used to obtain software or online services, which could be obtained more anonymously.

Furthermore this attack could also be used solely to damage the real customer by causing him to lose money or by ordering solely undesired goods or services for him that cause additional damage. The cause of additional, non-financial, damage through undesired service should be prevented by merchants and is not a specific problem of the payment system but of the identification of the orderer.

*Scale of Fraud:*

After stealing the personal data and setting up a customer entity multiple payments on the same customer’s account are possible. For that the prerequisite attacks P2, P3, P4 and P5 are needed only once. The purchase amount of one fake purchase request is only limited by the customer’s contract with the bank. Also the accumulated purchase amount per time (of multiple orders) and the maximum number of possible orders per time are only limited by means outside of the plain SET system.

*Indication of Fraud:*

The customer can detect the fraud by reading his account statement.

*Countermeasure a posteriori:*

The real customer must declare his signature key invalid at his certificate issuer.

### A2. SET Customer Impersonation without Signature Key

*Prerequisite Attacks:* P3 and P4 and P5

*Attack Description:*

This attack is the same as attack A1 but the signature key is not stolen. This means that the attacker must use the signature function in the customer's entity to sign data. The complete data structure to be signed, which is the PI-TBS structure, must be smuggled into the customer entity. Data which is already present there (e.g. the personal data) could be made available inside the customer entity after smuggling the other data elements to be signed into it, but this would require the construction of PI-TBS inside the customer entity and resembles more closely an attack as described in A4 (Additional Payment with Customer Entity). After signing the PI-TBS structure by applying the signature function SO in the customer entity, the signed data must be smuggled out of the entity and sent back to the attacker to construct the rest of the message.

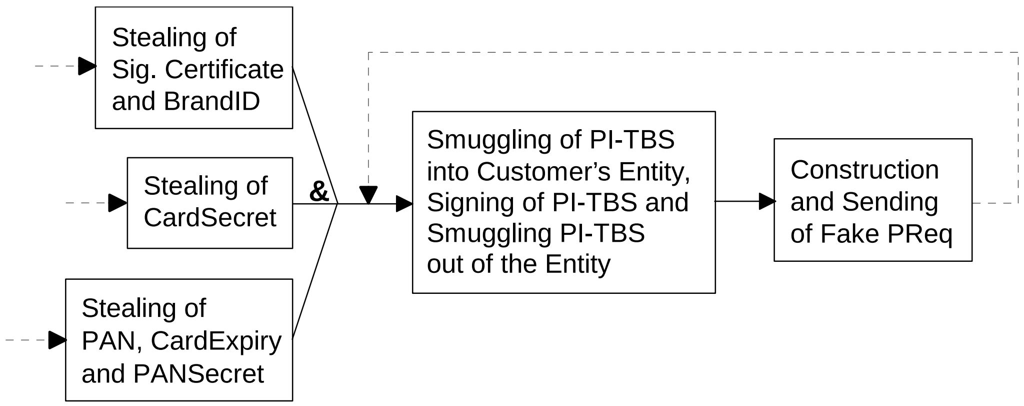


Fig. 4.13: SET Customer Impersonation without Signature Key

*Possible Intention of Attack:*

With this attack a payment at the expense of a particular other customer is possible, even if the signature key cannot be stolen. Otherwise the same is valid as for attack A1.

*Scale of Fraud:*

For one fraudulent payment the amount is limited only by general agreements outside of SET between the real customer and his bank, which are also valid for payments of the real customer. With this attack the signature function in the customer entity must be reused for each additional fraudulent payment. This means that repeated fraudulent payments are much more difficult than, for example, is the case for attack A1. Nevertheless the remaining personal data has to be stolen just once.

*Indication of Fraud:*

The customer can detect the fraud by reading his account statement. The smuggling of data into and out of the customer entity will not be realized by the customer with certainty.

*Countermeasure a posteriori:*

The real customer must declare his signature key invalid at his certificate issuer. But this prevents additional fraudulent payments only as long as no new customer's private signature key is installed. Moreover the customer cannot make payments either. If a new signature key is installed, the fraudulent person can continue making fraudulent payments without stealing the remaining private data again.

### A3. Fraudulent MOTO

*Prerequisite Attacks:* P4

*Attack Description:*

If an attacker has the name of a credit card owner, his private account number (PAN) and the corresponding expiration date of the credit card (CardExpiry) payments are possible.

For a conventional payment the credit card itself is the article that legitimatizes the card holder to make a payment. But payments are also possible with only the data embossed on the card. This data is the name of the credit card owner, the private account number (also called credit the card number) and the expiration date. Typically for orders where the customer is not personally present, e.g. mail or telephone orders, only these data are needed for a payment. This method of payment is called MOTO (**M**ail **O**rder/ **T**elephone **O**rder).

In the electronic SET system, the same data that is embossed on the plastic credit card is used for a purchase request of the customer. The private account number is populated in the data field PAN and the card expiry is populated in the field CardExpiry. This leads to the fact that after the data PAN and CardExpiry (attack P4) has been stolen and the name of the credit card owner has been obtained, fraudulent MOTOs are possible.

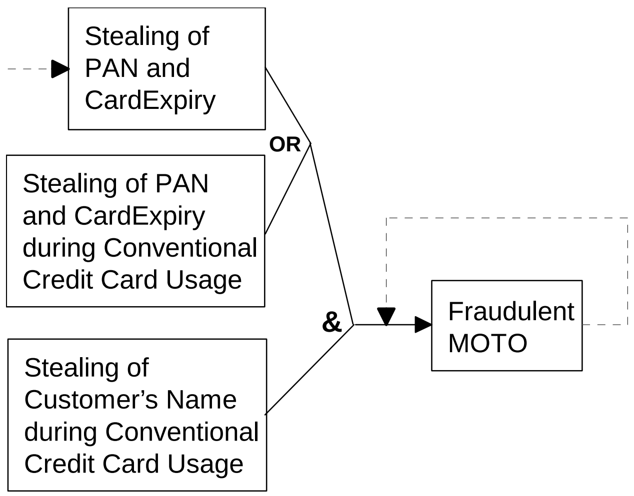


Fig. 4.14: Fraudulent MOTO

*Possible Intention of Attack:*

With this attack the obtaining of goods at the expense of another customer is possible. This goes for all goods that can be obtained via MOTO.

*Scale of Fraud:*

After the needed data has been stolen, multiple fraudulent orders on the same customer's account are possible. The prerequisite attack P4 is needed only once. Otherwise the same is valid as for attack A1. However, MOTO payments are not binding, i.e., it is the merchant's risk. This attack is expected to be ruled out by a change of the SET protocol in a future version (cf. chapter 5).

*Indication of Fraud:*

The customer can detect the fraud only by reading his account statement.

*Countermeasure a posteriori*

To stop further possible fraud the customer must block his private account number at his bank.

### A4. Additional Payment with Customer Entity

*Prerequisite Attacks:* P1

*Attack Description:*

The attacker uses the entity to make additional purchase requests that are not intended by the legitimate customer. The only difference between this and a normal purchase request is that the initiation for a request does not come from the legitimate customer but from a fake program which intrudes on the customer's entity (see prerequisite attack P1 for possible types of this program). This intruded program has to intercept input requests from the SET customer software to the customer and make the inputs itself. It has to simulate the legitimate customer. If the attacker intends not to reveal the attack to the customer, not only input requests from the SET software but all messages to the user interface have to be intercepted.

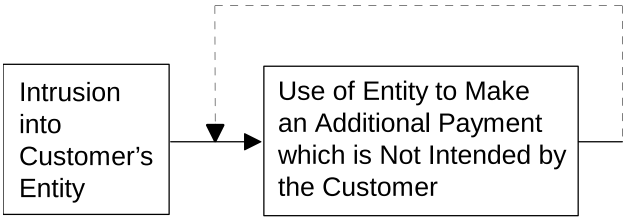


Fig. 4.15: Additional Payment with Customer Entity

*Possible Intention of Attack:*

The intention could be the same as in A1. With this attack nothing has to be stolen; more precisely, data must be smuggled out of the customer entity to the attacker. But the difficulty of this attack is to intercept the communication between the customer and his SET software.

*Scale of Fraud:*

For one fraudulent payment the same limits are valid as in A1. After the attack P1, multiple fraudulent payments are possible but not as easily as in A1 or A2 because after the intrusion of a program, the attacker has no direct control over it.

*Indication of Fraud:*

The customer can detect the fraud by reading his account statement. Furthermore a check of the software in the customer entity may detect the intrusion or intruded program.

*Countermeasure a posteriori:*

The intruded program must be deleted.

### A5. Payment of Unintended Purchase Amount

*Prerequisite Attacks:* P1
*Possible Fraudulent Person:* Merchant

*Attack Description:*

The purchase amount declared by the customer is included in the message PReq. There it is present as clear text and also in a hashed form. In clear text it is present in PurchAmt, an element of PI. In hashed form it is present in HOD. HOD is included in the PI and the OI.

PurchAmt as well as both HOD in PI and in OI are signed. If the merchant to whom the customer sends his purchase request is an attacker, it is possible that payment for an unintended purchase amount could be made by the customer.

An intruded program (attack P1) changes the element PurchAmt in HODInput directly before the hash HOD of HODInput is generated. Furthermore directly before the PI is signed it also changes its element PurchAmt. With these changes an unintended purchase amount is signed by the customer. Now the merchant only has to send an AuthReq based on the same fake purchase amount to the acquirer. The acquirer does not recognize any manipulation. He assumes that both are in agreement regarding the purchase amount. If the customer realizes the fraud later, e.g. by reading his account statement, he cannot prove that he intended to make a payment of a different amount, because his signature is decisive.

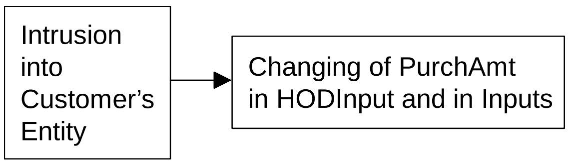


Fig. 4.16: Payment of Unintended Purchase Amount

*Variations:*

The actual amount of the payment to the merchant's account is based only on data sent from the merchant to the acquirer. The hash HOD in PI is used by the acquirer only to check the agreement between merchant and customer. The data PurchAmt in PI is not necessarily used by the acquirer before performing the payment transfer. This leads to the following variations.

Variation A: Manipulation of PurchAmt in only HODInput

If this attack variation is made, after a dispute the feasible check of the signature in the message PReq would reveal the inequality of the signed amount elements. Whether such an attack would nevertheless achieve a fraudulent payment is then dependent on the bank's procedure in this situation, which is not described in the SET specifications.

Variation B: Manipulation of PurchAmt only in PI

The manipulation of only the element PurchAmt in PI (more precisely of PurchAmt in Inputs) seems at first to be an attack, but it is not. After changing the PurchAmt in Inputs the fraudulent merchant would have only to change the data AuthReqAmt, which is the purchase amount in clear text (in the message AuthReq) on which the actual payment is later based. The hash in AuthReq would still be based on the real amount. The acquirer would not realize the fraud, because the hashes indicate agreement. If the customer realizes the fraud, e.g. by reading his account statement, after a dispute the signature check would reveal an inequality of the signed amount elements as in Variation A. In contrast to A, the inequality between amount information from the customer and the merchant cannot be recognized by the acquirer alone (the signature of the clear PurchAmt in PI can be checked by the acquirer alone), but only the customer can prove the inequality by disclosing the elements of HODInput. This makes the detection of fraud for this attack slightly more unlikely than in A (in A, an inconsistency could be revealed by the acquirer alone with an unusual check of the PurchAmt in PI, e.g. if he is suspicious of the merchant, but in B this is not possible). The decisive contrast to A is that after a dispute and a check of the signature, where also the customer is involved, it would be revealed that the merchant is the fraudulent person. The customer discloses the content of HODInput. The HOD of the customer and the merchant are the same. It follows from this that the merchant has used a different value for the PurchAmt which goes into his hash from the value for the AuthReqAmt.

Another variation could be that nothing is changed at the customer's side except the AuthReqAmt. Aside from the fact that an inconsistency would then be detectable by the acquirer alone, the same is valid as for the variation B.

*Possible Intention of Attack:*

A fraudulent merchant could attempt to attract customers with offers of low prices, and then bind them to higher prices in the order.

*Scale of Fraud:*

The limit for the fake purchase amount is the same as for the real purchase amount. This limit is defined by the contract between the customer and his bank and is not defined in the SET specification.

*Indication of Fraud:*

The customer can detect the fraud by reading his statement of account. The intruded program will not necessarily be detected by the customer. It could for instance delete itself after the manipulation and remain no traces. If the intruded program is used for manipulation of multiple payments and remains therefore in the customer entity, a check of the software may detect the intruded program. In the case of a non-modified attack and for variation B it is not possible for the acquirer to detect realizes the fraud alone.

*Countermeasure a posteriori:*

If the intruded program remains in the customer entity, it must be deleted.

### A6. Payment with Unintended Order Description

*Prerequisite Attacks:* P1
*Possible Fraudulent Person:* Chosen Merchant

*Attack Description:*

The description of an order is stored as the element OD in the structure HODInput. Not the data OD itself but its hash is a signed element of the PReq message. This signed hash is later compared by the acquirer with a hash generated by the merchant to check whether the customer and the merchant agreed on an order with the same description.

If an intruded program (attack P1) changes OD directly before the hash HOD is generated, a manipulated OD (precisely, its hash) will be signed later. If the merchant changes its OD equally, no mismatch will be recognized by the acquirer.

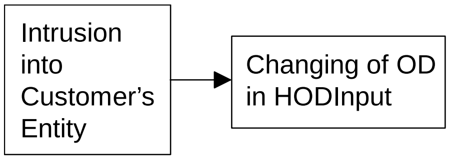


Fig. 4.17: Payment with Unintended Order Description

*Possible Intention of Attack:*

A fraudulent merchant could attempt to in advertise attractive descriptions in an offer but then change the order description in his own favor.

*Scale of Fraud:*

Since the purchase amount is not manipulated, the maximum scale of financial fraud is always limited to the purchase amount of the particular order. The intruded program can also be used to change descriptions of multiple orders. Changes are then possible whenever an order is made to the same merchant (the attacker) and the fake order description is roughly appropriate for the defined purchase amount.

*Indication of Fraud:*

The customer will realize the fraud when he receives the undesired commodity or service. For detection of the intruded program, the same is valid as for A5.

*Countermeasure a posteriori:*

If the intruded program remains in the customer entity, it must be deleted.

### A7. Payment to Unintended Merchant

*Prerequisite Attacks:* P1
*Possible Fraudulent Person:* Merchant

*Attack Description:*

The merchant to whom the customer intends to send his order is defined within the SET system solely by the element MerchantID in the message PReq. If this element is changed and all messages between the customer and the merchant are rerouted to a new merchant, the complete order and payment process will be performed with the new merchant.

For this attack all outgoing messages of the customer have to be rerouted to the new merchant. For this the intrusion of a program into the customer entity is necessary (attack P1). The moment messages have to be rerouted is after an offer has been selected, e.g. after a customer has browsed Web pages and selected goods, and before the first SET message is sent. This is the moment between stage 3 and 4 in the table in paragraph 2.1.2.

The MerchantID, which is part of the PI in the message PReq, has to be changed before the message is signed and decrypted. To prevent the customer from noticing of the different merchant identifier, the MerchantID has to be changed after its personal authentication, i.e. after it is displayed on the customer’s screen.

The following messages are then sent between the customer and the attacker as if the attacker were be the real merchant. Message requests for authorization and capture are sent from the attacker to his acquirer. At this point it can be seen that messages have to be rerouted before the first SET message is sent by the customer. Otherwise the customer would get the public encryption key of the real merchant’s acquirer, which would make it impossible for the attacker to forward the PI to his own acquirer.

The response message PReq (and also the optional InqReq) from the attacker back to the customer are signed. If the customer realizes the fraud as a result of the false signature, it is already too late to prove that he intended to order at another merchant, because the signature has been made with the new merchant’s (the attacker’s) identifier. If the attacker does not want the customer to realize the fraud on the basis of the response message, the intruded program which changes the MerchantID in the beginning also has to change a stored copy of this field which is then later be used to verify the signature of the response message. This works only if the verification of the response message does not include displaying the merchants name (whether this is included in the implementation is not specified in the SET specification). Otherwise the customer could see that the merchant he really ordered from is different from the merchant he intended to order from. An alternative for the attacker to prevent indication of fraud on the basis of the response message is to combine of this attack with attack A8.

Attacks A5, A6 and A7 are related. They all change some fields of an intended payment. As seen later they can all be prevented by a single method.

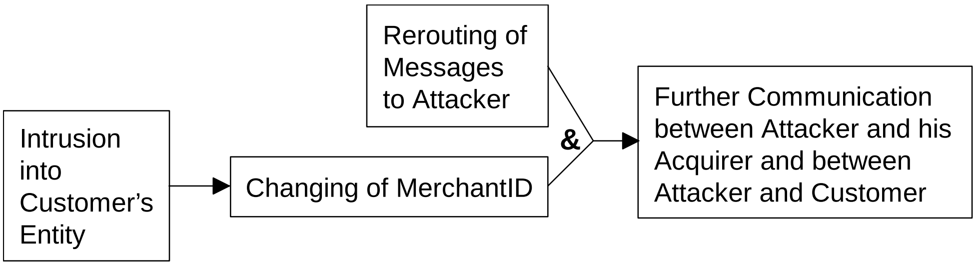


Fig. 4.18: Payment to Unintended Merchant

*Possible Intention of Attack:*

The intention of the attacker could be to misappropriate orders from another merchant who offers goods of the same kind as the attacker and who receives orders the attacker would like to have.

*Scale of Fraud:*

The order is sent to a merchant who is not the merchant the customer intended to order from, but the commodity or the service is delivered to the real customer. The scale of fraud for the customer is that the ordered articles or the service may differ from the desired products in terms of properties or quality. Further and considerable fraud is committed against the real merchant, who does not receive the order. The intruded program can also be used to reroute further orders. But only those orders for which the attacker knows the order description can be rerouted successfully and only for orders for goods he can deliver does the attacker receive payment.

*Indication of Fraud:*

In some cases the customer could realize the fraud by verifying the response message (see attack description). The customer could also realize the fraud after receiving the ordered commodity or service because of a difference in properties or quality. But in the case of this attack the customer does not necessarily realize the fraud at all.

*Countermeasure a posteriori:*

If the intruded program remains in the customer entity, it must be deleted.

### A8. Pretense of Correct Payment

*Prerequisite Attacks:* P6 or P7 or P8

*Attack Description:*

After one of the prerequisite attacks P6, P7 or P8, where the merchant's public signature key is changed, the attacker can send signed SET response messages to the customer and on the basis of the signature the customer does not recognize that the sender is a fraud.

With one of the prerequisite attacks the merchant's public signature key is changed into another public signature key to which the attacker has the corresponding private signature key. This key pair could be the real signature key pair of the attacker or it could be a newly generated key pair. The real attacker's signature key pair has the disadvantage that the attacker could be identified if fraud is detected (because of another attack) and the real certificate for the signature key of the attacker is found. It is possible to find the certificate because it contains the attacker's public signature key. With the certificate the attacker can then be identified.

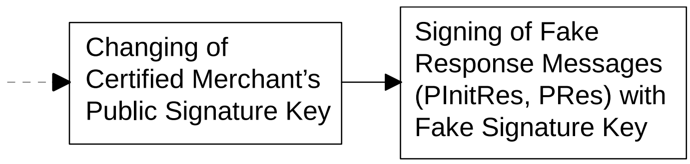


Fig. 4.19: Pretense of Correct Payment

*Possible Intention of Attack:*

This attack could be used together with attack A7 to prevent the detection of fraud by the customer.

*Scale of Fraud:*

Only as a result of this attack is no fraudulent payment transfer made. The damage made to the customer is that he believes the order was executed correctly, but it was not. If the order is not fulfilled correctly, he may wrongly accuse the real merchant of not filling the order properly.

*Indication of Fraud:*

The fraud will not necessarily be realized by the customer, for instance if the prerequisite attack P7 is used, where the merchant's public signature key is changed directly and the customer does not order again from the same merchant. Otherwise fraud will be recognized if the prerequisite attacks are detected (see P6 and P8).

*Countermeasure a posteriori:*

Depending on which prerequisite attack is used, the SET software must be reinstalled and the certificates and keys must be verified again.

### 4.3.3. Overview of Attacks

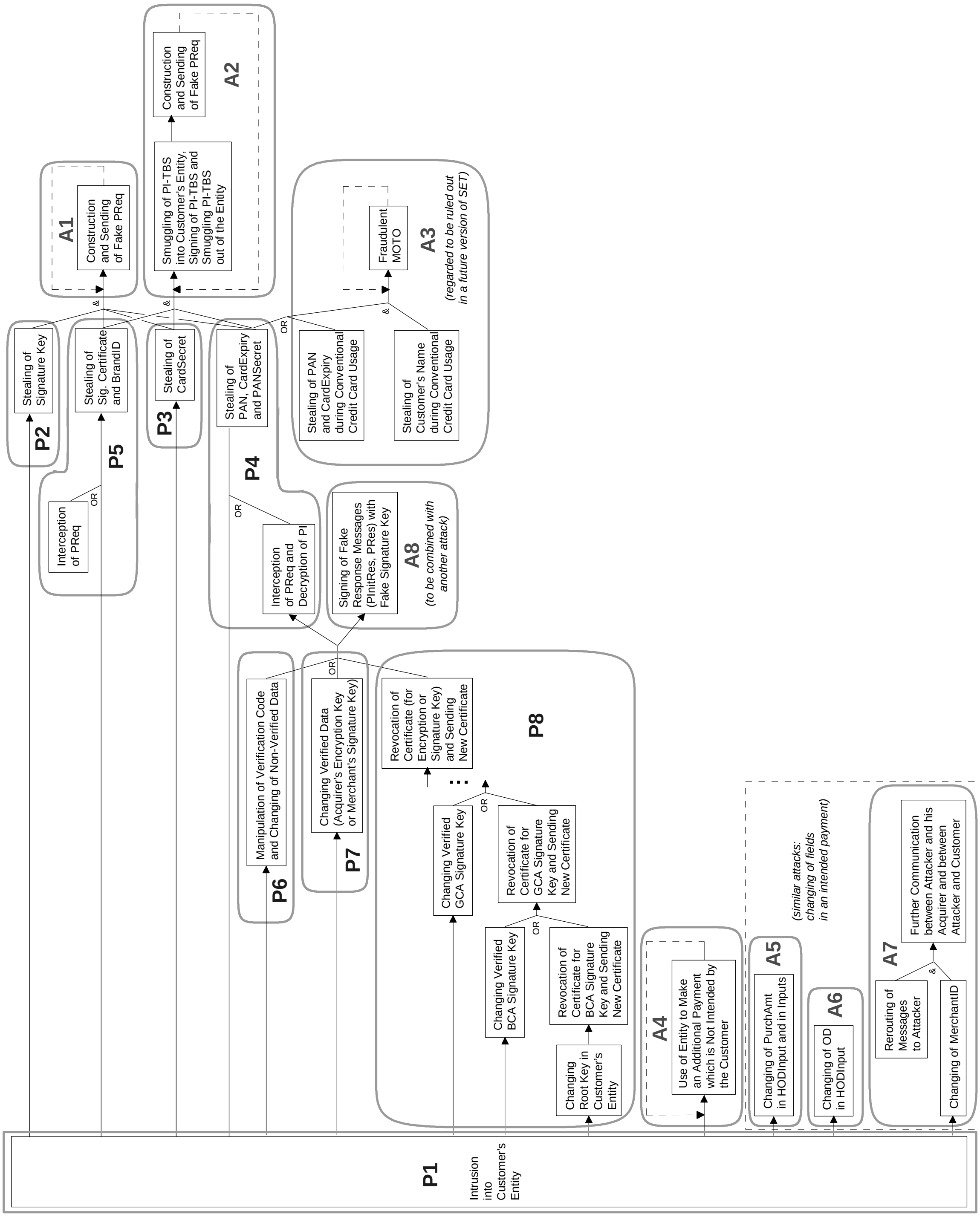


Fig. 4.20: Overview of Attacks

### 4.3.4. Scale of Fraud

The main attacks result in various scales of fraud to the customer. For a concrete attack against a definite customer the exact scale of fraud depends on further circumstances. The actual hardware and software implementation of the customer entity (different dependencies for A1, A2, A4) and the entire SET system is essential (different speed of multiple orders for A1, A2, A4). Furthermore the financial agreements are essential, which are for instance made inside the financial network, e.g. between the acquirer and the issuer, or between the customer and his bank (different amount and time limits for a payment transaction; essential for A1, A2, A4, A5). Also the kind of good ordered (essential for A6, A7) or the importance to the customer of receiving a response (essential for A8) affects the scale of the fraud.

Therefore a quantification of the attacks regarding the scale of fraud is not possible. Nevertheless a relative order can be given which is obviously somewhat subjective. For example multiple fraudulent orders without any customer's influence over them (A1) are certainly more undesirable than simply the manipulation of the purchase amount in one order (A5). Whether a merchant identity manipulation (A7) or a response information manipulation (A8) is more undesirable remains less obvious.

The different importance of preventing certain attacks can be deduced from the different scale of fraud. Hence the importance of protecting certain parts of the customer's SET system can be deduced. Attack A3 can be neglected because it is expected to be ruled out by a change of the SET protocol in a future version (cf. chapter 5).

| Attack | Scale of Fraud | |
|---|---|---|
| A1 | multiple fraudulent orders without dependency on customer entity | high |
| A2 | multiple fraudulent orders with partial use of customer entity | |
| A4 | multiple fraudulent orders with complete use of customer entity | |
| A5 | purchase amount manipulation in one order | |
| A6 | order description manipulation in one order | |
| A7 | merchant identity manipulation in one order | |
| A8 | response information manipulation for one order | low |
| [A3] | [multiple fraudulent orders without dependency on customer entity] | |

### 4.3.5. Needed Methods to Prevent Attacks

*Methods to prevent Attack A1:*

Attack A1 is only possible if all personal data can be stolen (see attack A1 for definition of personal data) . Therefore to prevent this attack only a single personal data element has to be protected against theft. The suitability of this data to be protected is different. Data that is not secret is not suitable. The account data is secret and strongly encrypted in messages, but data that is not sent at all is more suitable. This is the private signature key and the data CardSecret. Of these two elements, the private signature key is more suitable, because the customer entity is the only entity that knows it (the CardSecret is also known to the issuer). Attack A1 is prevented by method M1.

*Methods to prevent Attacks A2 and A4:*

In contrast to attack A1 in attack A2 the complete payment message is not generated outside of the customer entity. Some functionality is used inside the customer entity for every fraudulent payment again, particularly the signature function. This bypasses the protection of data, for instance the signature key, because the trusted device cannot distinguish between inquiries from the real customer and those from an attacker.

In attack A2 the complete functionality of the customer entity is not used to generate fraudulent payment messages. Take for instance the case where the CardSecret was previously stolen. A method similar to M1, where not only the signature key but also the CardSecret is secured, could prevent attack A2. But this is not a real advantage because more generally the attacker can use the customer entity for everything he is not able to steal. Furthermore, if an attacker has to use the customer entity for every fraudulent payment, it is possibly easier for him not to steal data beforehand but to use the entity to generate the complete fraudulent payment messages. Consequently a method to prevent attack A2 is only a real advantage if it also prevents attack A4. This is achieved by method M2.

Method M2 requires a trusted input device (cf. section 3.3). This makes method M2 essentially more expensive to realize than M1. Methods M3, M4, M5 and M6 do not require a trusted input device. They do not prevent attack A4 (and hence A2) completely, but nevertheless they make them a much more difficult or at least increase the probability of detection.

*Methods to prevent Attacks A5, A6 and A7:*

Attacks A1, A2 and A4 generate additional payments which are not intended by the customer. In contrast to these the attacks, A5, A6 and A7 manipulate intended messages. If the content of an intended message is manipulated, a general confirmation of the message without the personal check of its content (method M2) does not prevent the attack. Method M7 is an extension of method M2. It prevents attacks A5, A6 and A7.

*Methods to prevent Attack A8:*

With method M9 the signature verification part is secured. Therefore it prevents attack A8.

*Methods to prevent Attack A3:*

Method M8 prevents attack P4. Hence it also prevents attack A3, because P4 is a prerequisite attack for attack A3. But as described in paragraph 4.3.4 a prevention of attack A3 is not considered important because it is expected to be ruled out in a new version of SET.

## 4.4. Methods

In this section for the different security critical parts of the customer’s SET software system (see section 4.2) implementation methods are developed. Methods for signature generation are developed in paragraph 4.4.1. These methods are further discussed in paragraph 4.4.2 and 4.4.3. Methods for encryption and signature verification are developed in paragraph 4.4.4, and methods for mobility are developed in paragraph 4.4.5.

### 4.4.1. Methods for Signature Generation

A signature with the private key of the customer is required by the SET specification for the messages PReq and InqReq. The following methods are explained with the example of the message PReq, but their concepts can also be used for the message InqReq, if necessary.

#### M1. Signature Key and RSA Algorithm on Trusted Device

The benefit of this method is to prevent the customer’s private SET signature key from being stolen. Not only the key is stored on the trusted device but also the algorithm that uses it, the RSA signature algorithm. As a result, the key does not leave the trusted device and is protected from theft. This method prevents the prerequisite attack P2 and hence attack A1.

Only a small part of the signature generation (see fig. 4.2) is implemented on the device as shown in the following figure.

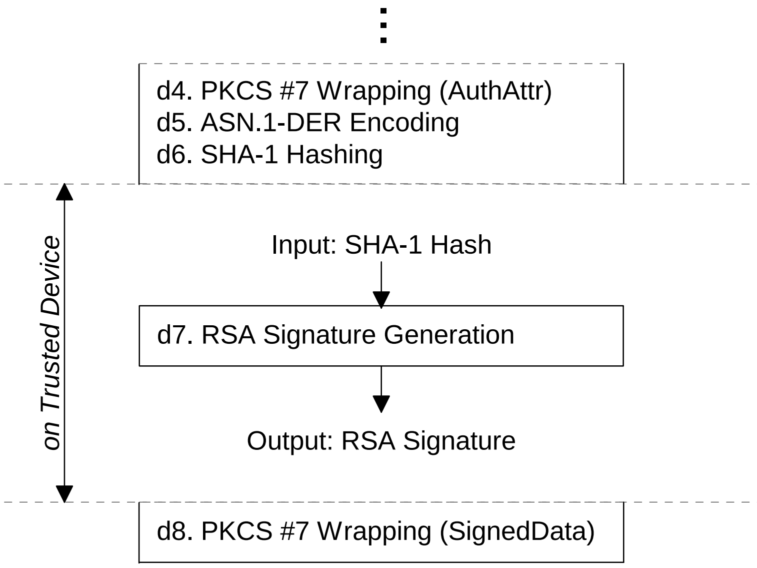


Fig. 4.21: Procedure for Method M1

Only little functionality and data are required on the trusted device.

| *Needed Functionality* |
|---|
| • RSA Signature Algorithm |

| *Needed Data* |
|---|
| • Private Signature Key |

### M2. Personal Confirmation on Trusted Device

*Special Requirement on Trusted Device:*

- Trusted User Control (requirement iii., see section 3.3)

The benefit of this method is that signature generation is only possible if the customer is personally present and intends to make a signature. It prevents attacks A1, A2 and A4.

The trusted device requests input from a person before the RSA signature (step d7 in fig. 4.2) is generated. As described in section 3.3 the confirmation can be the pressing of a single button "yes". To ensure that no signature generation is possible without confirmation by a person, the private signature key and the RSA function must be stored on the trusted device. Hence method M2 is an extension of the method M1.

Only little functionality and data are required on the trusted device.

| *Needed Functionality* |
|---|
| • Wait for Input "yes" from Trusted User Control<br>• RSA Signature Algorithm |

| *Needed Data* |
|---|
| • Private Signature Key |

## M3. Logging of Data on Trusted Device

*Special Requirement on Trusted Device:*

- Trusted Output to the User (requirement iv., see section 3.3) at any time after signature generation

With this method the trusted device accesses the data to be signed before it is hashed. This enables a logging of the data in clear text on the device and a later check of the data when a trusted output device is available.

The following figure shows which procedure is to be implemented on the untrusted workstation and which procedure is to be implemented on the trusted device.

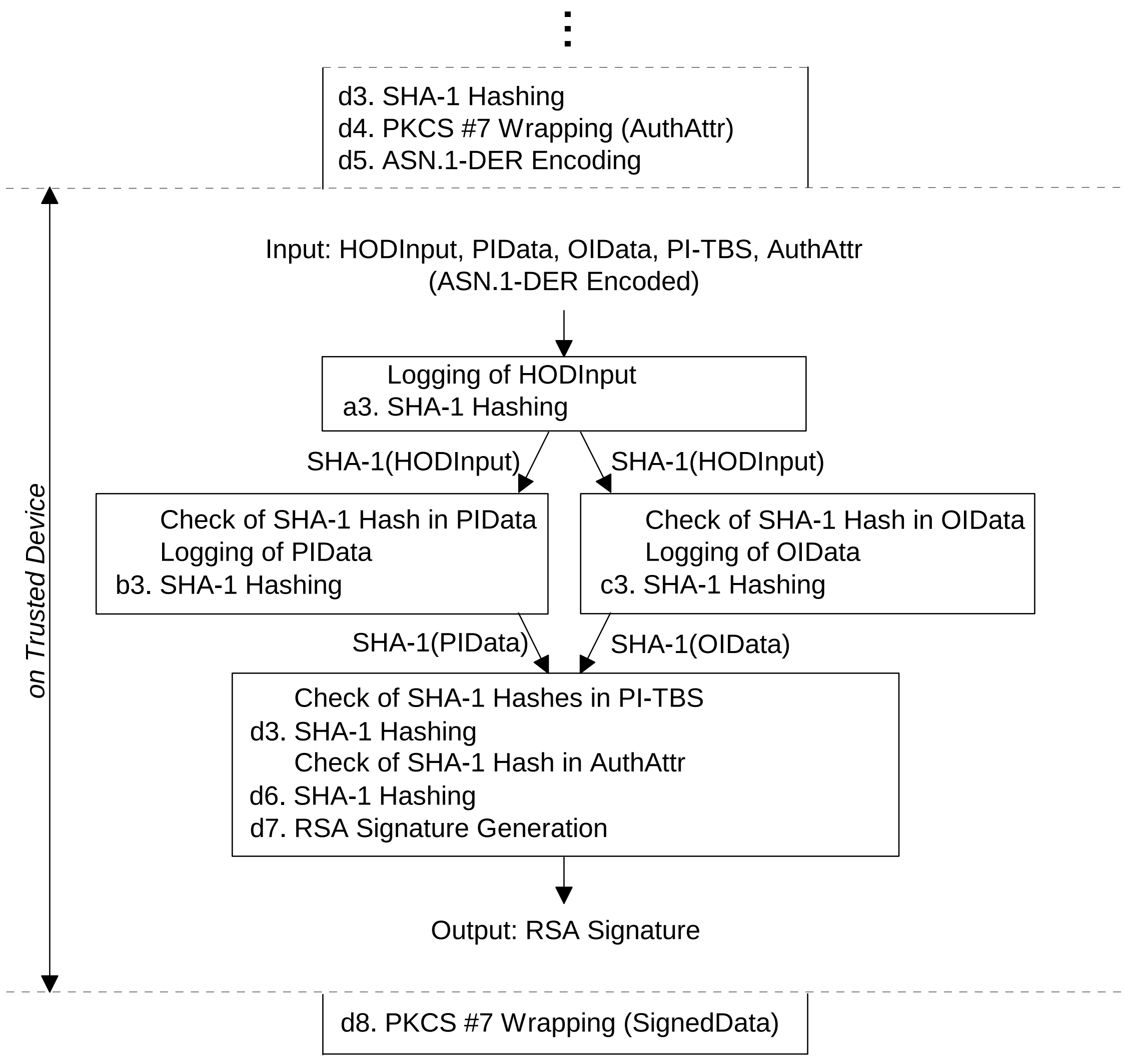


Fig. 4.22: Procedure for Method M3

The procedure for the untrusted workstation is the same as in figure 4.2 up to step d5. The input for the trusted device are all data structures to be hashed. They are already ASN.1-DER encoded. The hash SHA-1(HODInput) included in PIData and OIData, the hashes

SHA-1(PIData) and SHA-1(OIData) included in PI-TBS and the hash SHA-1(PI-TBS) in AuthAttr have to be checked to ensure that these are really the hashes of the previously logged data. The hashes are generated by the device itself and afterwards simply compared with the hash elements in the input. As in method M1, step d8 is not implemented on the trusted device but on the workstation.

The following table summarizes the functionality required on the trusted device.

| *Needed Functionality* |
|---|
| • Logging of Elements<br>• SHA-1 Hash Algorithm<br>• Check of Hashes with Previously Generated Hashes<br>• RSA Signature Algorithm<br>• Output of Logged Data (if requested) |

Furthermore the following data, which is not input data, must be stored on the trusted device.

| *Needed Data* |
|---|
| • Private Signature Key |

With the logging function particular elements can be copied from the input and stored on the trusted device, for instance the elements PurchAmt in HODInput, PurchAmt in PIData and MerchantID in PIData. If the storage capacity is big enough, the OD in HODInput or possibly even the complete input can also be stored. The logging of this data has the advantage that it is later possible for the customer to check what was really signed. Nevertheless this is only useful if at some time after the execution a trusted output is feasible. For example this is the case if the device is mobile and one of the following trusted output devices exist.

- *Output Device at Customer's Home*
- *Unlinked Output Device*
  This output device might, for example, be another computer. It is trusted, because it has no link to the entity whith which the payment was made. If data is logged and afterwards displayed on another computer, it is unlikely that the same attacker would be able to manipulate the output on the second computer such that the customer would not detect the attack. Nevertheless such a manipulation cannot be ruled out completely.
- *Mobile Output Device*
  This device is trusted because it is owned and carried exclusively by the customer. It might, for example, be a simple reader with a display. After the signature is made the device with the logs is disconnected from the customer entity and connected to the mobile output device.

### M4. Predefinition of Limits of Trusted Device

*Special Requirement on Trusted Device:*
- Trusted User Input (requirement iii., see section 3.3) at any time before signature generation

This method equals method M3 with the difference that the data is not logged but checked. If a trusted input device is available at any time before a signature is generated, it is possible to store some values on the device which can later be used to check the data to be signed. Suitable values are listed in the following.

| *Signature Limits* |
|---|
| • Limit for Accumulated Number of Signatures<br>• Limit for Number of Signatures per Time<br>• Limit for Number of Signatures per Device Use |

| *Purchase Amount Limits* |
|---|
| • Limit for Purchase Amount per Signature<br>• Limit for Accumulated Purchase Amount<br>• Limit for Purchase Amount per Time<br>• Limit for Purchase Amount per Device Use |

| *Restriction of Merchants* |
|---|
| • Allowed MerchantID's<br>• Not Allowed MerchantID's |

| *Date Limit* |
|---|
| • Deadline for Purchase Date |

If the limit of an accumulated value is reached (number of signatures or purchase amount) or the date (PReqDate) exceeded the deadline, no further signature is made by the device unless the values are newly defined. To prevent manipulation of these values by an attacker, changing them can be protected with a special password or PIN.

The limits per time require a trusted device with a clock. A real clock on a trusted device is only possible if the device has a constant power supply. This is not the case with simple devices for only processing and data storage, in particular mobile devices without a battery. An alternative could be a time counter that simply counts while the device is supplied with power. Another alternative for mobile devices is to define of a time limit per device use. The device use time means the time between connecting and disconnecting the device with the workstation (the device can consider this time as the time it is supplied with power).

The following table summarizes the functionality required on the trusted device.

| *Needed Functionality* |
|---|
| • Protected Mode for Predefining Limits<br>• Check of Elements with the Help of Predefined Limits<br>• SHA-1 Hash Algorithm<br>• Check of Hashes with Hashes Generated Before<br>• RSA Signature Algorithm |

Furthermore the following data, which is not input data, must be stored on the trusted device.

| *Needed Data* |
|---|
| • Predefined Limits<br>• Private Signature Key |

### M5. Predefinition of Constants on Trusted Device

*Special Requirement on Trusted Device:*
- Trusted User Input (requirement iii., see section 3.3) at any time before signature generation

This method equals method M3 with the difference that the data is not logged but constant elements are checked. As in method M4 a trusted input device must be available before a signature is generated. The following table lists the elements that remain unchanged for different transactions.

| *Constant Elements* |
|---|
| • PaySysID<br>• Language<br>• SWIdent<br>• PAN (BIN)<br>• CardExpiry<br>• PANSecret<br>• BrandID |

The BrandID is a constant element if the customer is a customer of only one brand. The constant elements are checked either by comparing them with the predefined values or by populating their corresponding fields with predefined values. The following table summarizes the functionality required on the trusted device.

| *Needed Functionality* |
|---|
| • Protected Mode for Predefining Constant Elements<br>• Check of Constant Elements<br>• SHA-1 Hash Algorithm<br>• Check of Hashes with Hashes Generated Before<br>• RSA Signature Algorithm |

Furthermore the following data, which is not input data, must be stored on the trusted device.

| *Needed Data* |
|---|
| • Predefined Constant Values<br>• Private Signature Key |

### M6. Generation of Elements on Trusted Device

*Special Requirement on Trusted Device (for Generation of PReqDate):*
- Permanent Power Supply

The following elements can be generated by a trusted device itself. The benefit of this method is that these elements can be trusted by the customer and do not have to be checked by other means.

- XID
- Chall-C
- ODSalt
- PReqDate
- HOD
- TransStain

The procedure needed for this method is the same as for M3 or M4 with the difference that instead of logging the elements or comparing them with a limit, the corresponding fields in PIData, OIData or HODInput are populated by the device. If a PInitReq/Res message pair is used for the transaction, the elements XID and PReqDate are generated by the merchant and this method can not be used for these elements.

The following table summarizes the functionality required on the trusted device. For random number generation see [MaVi97b, p.67,525]; for the keyed-hash operator HMAC see [MaVi97b, p.96].

| *Needed Functionality* |
|---|
| • Random Number Generation (for XID, Chall-C, ODSalt)<br>• Clock (for PReqDate)<br>• HMAC Keyed-Hash Algorithm (for TransStain)<br>• SHA-1 Hash Algorithm<br>• Check of Hashes with Hashes Generated Before<br>• RSA Signature Algorithm |

Furthermore the following data, which is not input data, must be stored on the trusted device.

| *Needed Data* |
|---|
| • Random Seed or Pool (for Random Number Generation)<br>• CardSecret (for TransStain)<br>• Private Signature Key |

### M7. Personal Check of Elements on Trusted Device

*Special Requirements on Trusted Device:*

- Trusted Output to the User (requirement iv., see section 3.3)
- Trusted User Control (requirement iii., see section 3.3)

The benefit of this method is that the customer can check important elements personally before signing them. The procedure for this method is the same as for method M3 with the difference that the elements are not logged but important elements are extracted for checking. Which elements are important elements to be checked personally is discussed in the next section. For example at least the PurchAmt should be checked with this method. The procedure shown in the next figure is an example of this method, where the important data are the elements PurchAmt, OD and MerchantID.

The MerchantID can not be checked by the customer directly. It is only a number, and if only this number is the output to the user of the trusted device, it is not possible for the customer to link this number to the real name of the merchant. The name is used in the following as the piece of information in clear text which identifies the merchant. It could also include his address or something else, but it must be used in the merchant’s certificate. In this method the merchant’s name is extracted from the merchant’s certificate by the untrusted workstation. The name is the additional input to the trusted device (not shown in the figure). Instead of the MerchantID, the received name in clear text is checked. The trusted device adds the name as a PIExtension to the PIHead data structure. With this method, in the event of a subsequent dispute, the customer can prove which was the intended merchant’s name. However, the meaning (syntax and semantic) of this extension must be defined beforehand. If for SET unique merchant names are defined, these names can completely replace the purpose of the MerchantID.

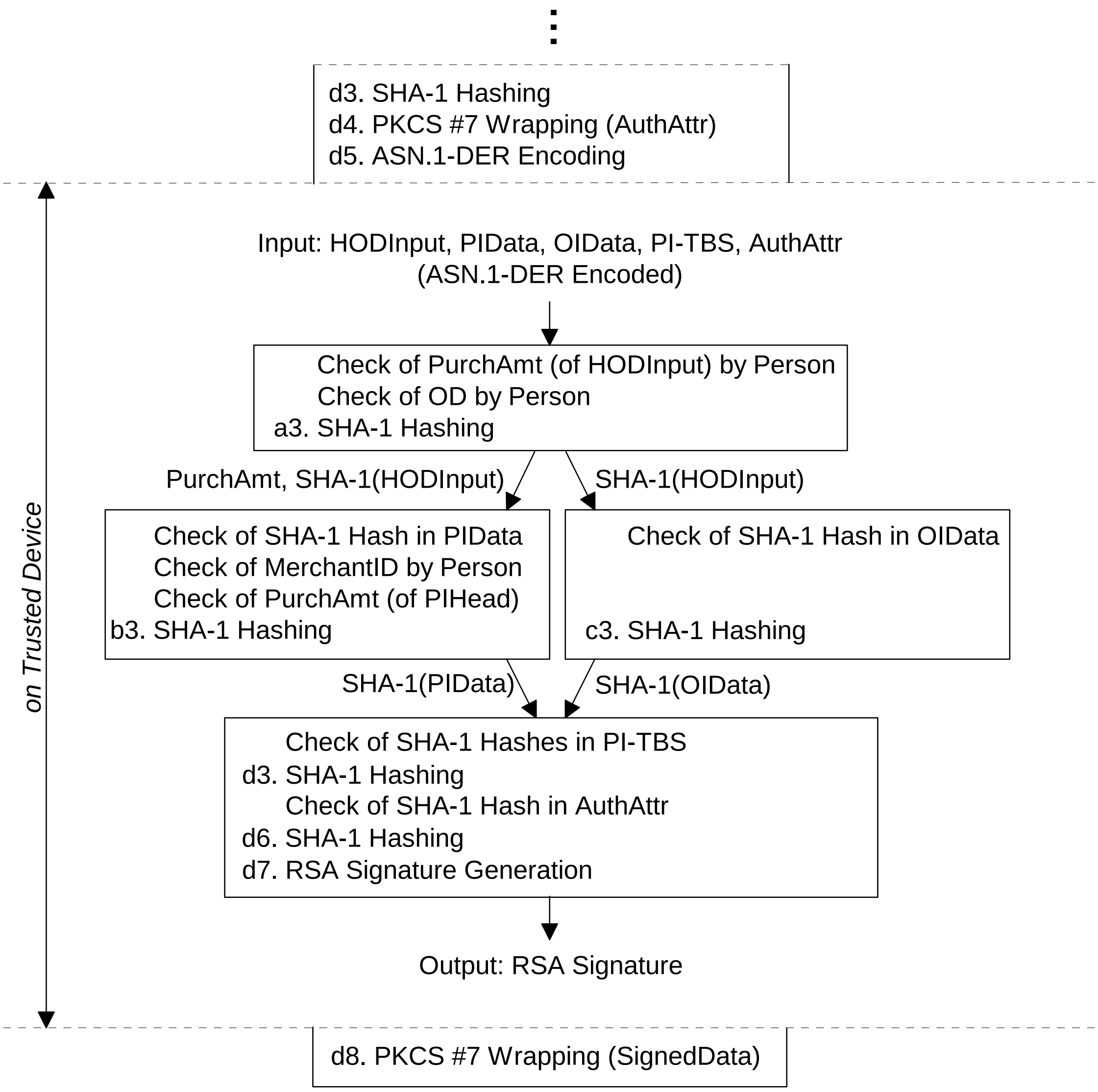


Fig. 4.23: Procedure for Method M7

The extracted data is sent to the user output facility. With the output facility the customer can check the data personally. Only if the data is correct to his mind does he confirm it by pressing the key "yes" and only then is a signature made. The PurchAmt of PIHead (which is part of PIData) must be set to the same value as the PurchAmt in HODInput. Hence this the PurchAmt of HODInput is stored after it is checked by the person and later compared with the value in PIHead (step above b3).

The requirements of the elements on the output device, for instance a display, are different. For example the PurchAmt consists only of a short character sequence and therefore the requirements are simple. In contrast the length of, for example, the order description (OD) is not limited by the SET specification. The following table summarizes the functionality required on the trusted device.

| *Needed Functionality* |
|---|
| • Personal Check of Elements<br>• SHA-1 Hash Algorithm<br>• Check of Hashes with Hashes Generated Before<br>• RSA Signature Algorithm |

Furthermore the following data, which is not input data, must be stored on the trusted device.

| *Needed Data* |
|---|
| • Private Signature Key |

### 4.4.2. Variations of Methods M3, M4, M5, M6 and M7

All methods logging data (M3) or checking data (M4-7) require data structures as input which contain elements to be checked. In the last paragraph each of these methods is described with all existing structures as input. These input structures are HODInput, PIData, OIData, PI-TBS and AuthAttr. But it may not be necessary to input all these structures. This paragraph describes possible subsets of structures as input and defines variations of the methods M3 to M7.

If only a subset of structures are input, part of the message remains unchecked. To determine which part must be checked, the elements need not be examined but rather the different instances of these elements which enter into the message. For example the value of the element PurchAmt enters into the message six times. It is contained in two different structures, in HODInput and in PIData. The structure HODInput enters into the message four times and the structure PIData twice. The value of PurchAmt when entering into the message is called an instance here. Therefore the PurchAmt has six instances. These instances may have different values if they enter into the message on the untrusted workstation.

Which structures are chosen to be input for a method depends on which instances must be checked. The following figure illustrates the different instances. This figure follows from the simplified data structure of the message PReq in figure 2.7.

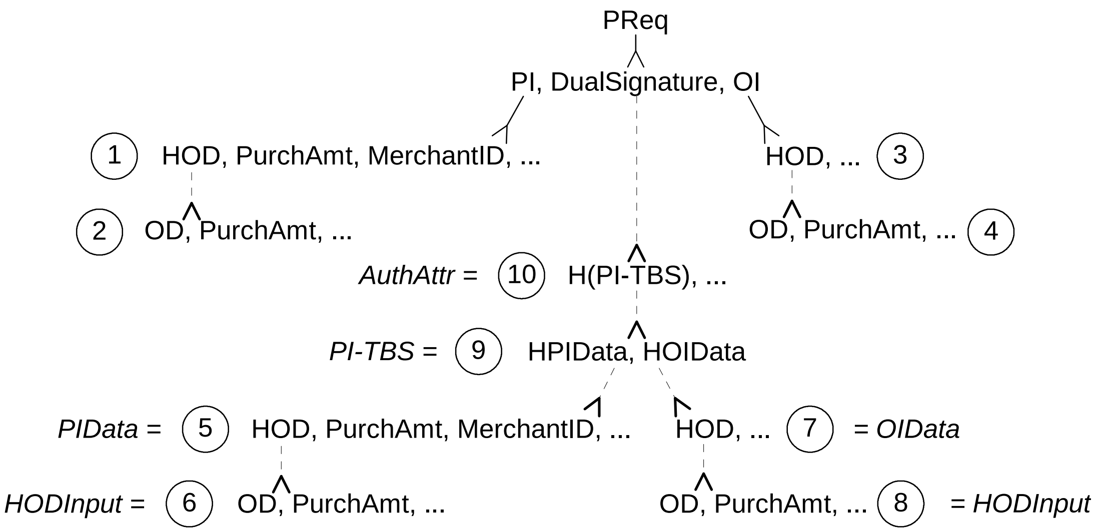


Fig. 4.24: Different Instances of Elements Entering Into PReq

The instances entering into the PI or the OI need not be checked (instances marked 1 to 4), because their integrity and data origin authenticity is guaranteed by the signature. For example, if the MerchantID in the PI is considered an important element, not the instance marked 1 has to be checked but rather the corresponding instance marked 5. Incidentally, as a result, the construction and encryption of the PIData (this is more precisely the function EX(P, PI-OILink, PANData), see fig. 2.6) as well as the construction of the OIData can be done by the untrusted workstation without consequences for integrity and data origin authentication of the elements.

The signature binds the signed data to the private key. The signed data is the structure AuthAttr. This structure contains a hash of a structure which in turn contains hashes of further structures, etc. (see fig. 4.24). Hence, if instances of a lower structure shall be checked, the hash of this lower structure and the hashes of all higher structures must also be checked on the trusted device. An instance which is a hash value is checked by generating the hash on the trusted device and comparing it with the instance.

Therefore, if instances of a structure shall be checked, this structure and all higher structures must be input to the trusted device. Otherwise the binding of the checked instances to the private key would get lost. For example, if instances marked 5 shall be checked on the trusted device, the structures marked 5, 9 and 10 must be input to the trusted device. Hence the variations listed in the following are all theoretically possible. As in method M7, the input data is ASN.1-DER encoded. The corresponding procedures are analogous to the procedure of M7.

| | |
|---|---|
| *Variation a* | Input: PIData, PI-TBS, AuthAttr<br>Not Checked: Instances in 6 (HODInput), 7 (OIData) and 8 (HODInput) |
| *Variation b* | Input: HODInput, PIData, PI-TBS, AuthAttr<br>Not Checked: Instances in 7 (OIData) and 8 (HODInput) |
| *Variation c* | Input: OIData, PI-TBS, AuthAttr<br>Not Checked: Instances in 5 (PIData), 6 (HODInput) and 8 (HODInput) |
| *Variation d* | Input: HODInput, OIData, PI-TBS, AuthAttr<br>Not Checked: Instances in 5 (PIData) and 6 (HODInput) |
| *Variation e* | Input: PIData, OIData, PI-TBS, AuthAttr<br>Not Checked: Instances in 6 (HODInput) and 8 (HODInput) |
| *Variation f* | Input: HODInput, PIData, OIData, PI-TBS, AuthAttr<br>Not Checked: - |

The methods described in paragraph 4.4.1 correspond to variation f. The benefit of the other variations is that they require less resources of the trusted device, a less complex procedure and less time to generate the signature. Low requirements placed on the resources of a trusted device are important if cheap trusted devices, especially smartcards, shall be used. The disadvantage of variations a to e is that they possibly achieve less security (this is discussed in the following). Whether they constitute a real advantage must be considered carefully. Therefore for each instance of the elements in message PReq it is determined in the following paragraph whether and with which method this instance should be checked. Thus possible security problems of variations a to e can be determined.

### 4.4.3. Signature Methods Needed for Instances

In the SET specification the entire workstation is considered a trusted device. Because the resources of the workstation are sufficient, security features, e.g. as given by the signature in the message PReq, can be provided for a number of structures as a whole. This can lead to implementation problems if such features shall be realized with small trusted devices like smartcards which have limited resources.

As described in the previous paragraph with the example of the signature generation for the message PReq, enough security might be achieved if only a subset of structures is given to the trusted device as input. For example several elements are multiply included as different instances in different structures. It may be sufficient if only one instance of an element is signed.

Different instances require different security features. For example the XID achieves a signature uniqueness to prevent replay attacks. It must be a unique value and therefore can not be checked personally. Because a non-unique value, more precisely a value already used for a different transaction, would only abort the transaction processing, the value need not be generated on a trusted device. The confirmation of signature generation (with an unknown but definitive value) is sufficient to prevent replay attacks (achieved by method M2). In contrast the PurchAmt, for example, should be checked personally before it is signed to ensure that it is really the value intended by the customer (achieved by method M7).

In the following it is determined which signature methods are needed for which instances. Whether the variations described in the last paragraph are practically usable for implementations is discussed regarding consequences on security afterwards.

| **PIData** | | |
|---|---|---|
| *Name of Instance* | *Proposed Method* | *Possible Consequences if Method is Not Used* |
| LID-C, [LID-M] | - | - |
| XID | M2 | -no prevention of replay attacks on PIData [cf. MaVi97b, p.70,451] |
| PReqDate | M6 or M7 | -PReqDate not provable by customer |
| [PaySysID] | - | - |
| Language | M5 | -customer does not understand clear text in response messages from acquirer |
| HOD | M6 | -intended purchase amount, order description, recurrences and installments (if used) not provable by customer |
| PurchAmt | M7 | -intended purchase amount not provable by customer |
| MerchantID | M7 (with name in clear; see M7) | -intended merchant not provable by customer |
| [InstallRecurData] | M7 | -intended recurrences or installments not provable by customer |
| TransStain | M6 | -CardSecret no longer secret<br>-customer cannot give evidence to his issuer that he participated in a particular transaction [cf. MaVi97b, p.56] |

| | | |
|---|---|---|
| [SWIdent] | M5 | -fault caused by security flaw of software not provable by customer |
| [AcqBackKeyData] | M5* | -no secrecy of AcqCardMsg (see fig. 2.9) |
| [PIExtensions] | dependent on particular extensions | |
| PAN, CardExpiry | M5 | -false value causes abortion of transaction |
| PANSecret | M5* | -false value causes abortion of transaction<br>-no prevention of guessing attacks on PAN in customer certificate [cf. MaVi97b. p.283] |
| EXNonce | M6* | -no prevention of dictionary attacks on PANData [cf. MaVi97b. p.283] |

*This method is only useful if the encryption functionality is also implemented on the trusted device.

| **OIData** | | |
|---|---|---|
| *Name of Instance* | *Proposed Method* | *Possible Consequences if Method is Not Used* |
| LID-C, [LID-M] | - | - |
| XID | M2 | -no prevention of replay attacks on OIData [cf. MaVi97b, p.70,451] |
| PReqDate | M6 or M7 | -PReqDate not provable by customer |
| [PaySysID] | - | - |
| Language | M5 | -customer does not understand clear text in response messages from merchant |
| RRPID | - | - |
| Chall-C | M6 | -freshness of PRes can not be checked |
| HOD | M6 | -intended purchase amount, order description, recurrences and installments (if used) not provable by customer |
| ODSalt | M6 | -false value causes abortion of transaction |
| [Chall-M] | M6* | -false value causes abortion of transaction |
| BIN | M5 | -false value causes abortion of transaction |
| [ODExtOIDs, OIExtensions] | dependent on particular extensions | |

*Use of method M6 means here the signature verification of message PInitRes. Chall-M is not generated on the trusted device but securely received from the merchant. Therefore this method is only useful if the signature verification functionality is also implemented on the trusted device.

| **HODInput*** | | |
|---|---|---|
| *Name of Instance* | *Proposed Method* | *Possible Consequences if Method is Not Used* |
| OD | M7 | -intended order description not provable by customer |
| PurchAmt | M7 | -intended purchase amount not provable by customer |
| ODSalt | M6 | -no prevention of dictionary attacks on HOD [MaVi97b, p.326] |
| [InstallRecurData] | M7 | -intended recurrences and installments not provable by customer |
| [ODExtensions] | dependent on particular extensions | |

*The table does not differentiate between instances for HOD in PIData and instances for HOD in OIData because the respective consequences are the same.

The consequence "not provable by customer" is given for each instance that belongs to an element of which a number of instances enter into PReq. In other words, if a number of instances of an element are signed (in a single PReq) but the instances have different values, the customer cannot prove which value is the intended one. For example, if the instance of PurchAmt entering into structure 5 (see fig. 4.24) differs from the instance entering into structure 6, no rules are defined in the SET specification concerning the consequences (it is not checked even if the values are the same). Therefore for this work it is assumed that in that case the purchase request becomes invalid.

Therefore if the intended PurchAmt shall be provable, the instance in HODInput both for PIData and OIData must be checked. As described in paragraph 4.4.2 all higher structures must then also be checked. The result is that variations a to e should not be used for implementations.

The second column of the tables propose methods to be used for the instances. These proposals are used later for the implementation proposals (4.5), which are combinations of methods. For all instances for which method M7 is proposed, alternative methods are M3 and M4. These methods should be used if method M7 cannot be implemented.

### 4.4.4. Methods for Encryption and Signature Verification

#### M8. Delegation of Encryption

*Special Requirement:*

- Trusted Extra Device for Encryption (Stationary)

The benefit of this method is that data confidentiality features are achieved without implementation of the SET encryption functionality on the (possibly mobile) device. As described in section 4.2 an implementation of that functionality is not achievable for trusted devices with low resources, because this would also require the implementation of the certificate verification functionality on the same device.

The idea of this method is that less encryption functionality is needed on the trusted device if the message is encrypted for and sent to an extra device. This device decrypts the message (therefore it must be a trusted device), encrypts it for the merchant and sends it to him. The trusted extra device needed for the SET encryption functionality can be for example a private stationary hardware device connected to the Internet and dedicated to the purpose of this method.

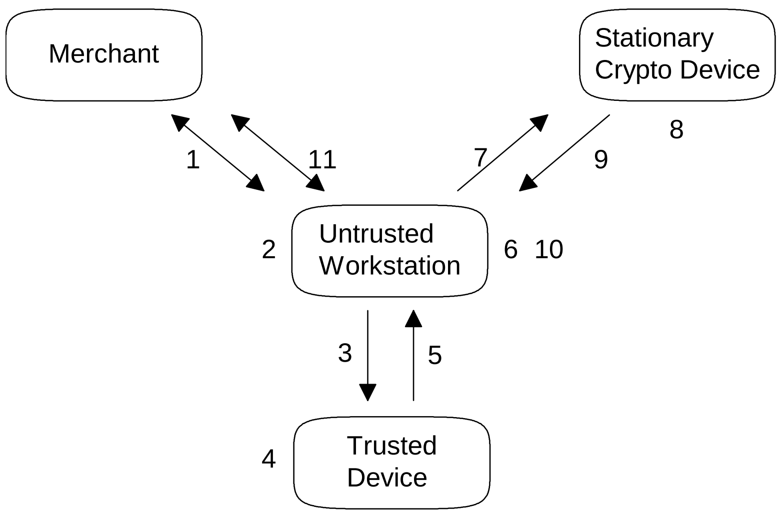


Fig. 4.25: Delegation of Encryption

The encryption algorithm on the trusted device can be asymmetric or symmetric. The advantage of an asymmetric algorithm is that if the RSA signature algorithm is also implemented on the device, the same algorithm can be used for signing and encryption. In this case, not the signature key but an extra encryption key is used for encryption and only the stationary crypto-device possesses the corresponding decryption key. The advantage of a symmetric algorithm is that this algorithm is faster.

The encryption keys can be generated by the stationary crypto-device. The encryption key can be stored on the trusted device when a trusted connection is available, for example the trusted device and the stationary device can be plugged together before any purchases are made with the trusted device.

The only data that need to be encrypted by the trusted device is the PIData in the message PReq. With this method the following message flow between the trusted device, the crypto-device and the merchant is given.

| | |
|---|---|
| Step 1 | beginning of SET processing |
| Step 2 | generation of structure PIHead |
| Step 3 | sending of PIHead to trusted device |
| Step 4 | appending of PANData to PIHead; encryption of PIData |
| Step 5, 6, 7 | sending of encrypted PIData via workstation to crypto device |
| Step 8 | decryption of PIData; generation of SET's EX structure |
| Step 9 | sending of EX structure to workstation |
| Step 10 | construction of PReq message |
| Step 11 | further SET processing |

With this method the confidentiality of the PANData, which is stored on the trusted device, is preserved and possibly that of other payment information generated by other methods on the trusted device. The confidentiality of payment information exchanged during the initiation process, for instance the order description, cannot preserved with this method. The following table summarizes the functionality required on the trusted device.

| *Needed Functionality* |
|---|
| • Encryption Algorithm (RSA or DES) |

Furthermore the following data, which is not input data, must be stored on the trusted device.

| *Needed Data* |
|---|
| • PANData<br>• Encryption Key |

### M9. Delegation of Signature Verification

*Special Requirement:*

- Trusted Extra Device for Signature Verification (Stationary)

The idea behind this method is similar to that behind the previous method. The data signed by the merchant is sent to the crypto device. The crypto device verifies the signature and replaces the signature with its own signature. The newly signed data is then sent via the workstation to the trusted device. Although the merchant's signature is replaced, its freshness can still be checked on the trusted device. For this the customer's challenge value (Chall-C in PReq or Chall-C2 in InqReq) must be generated on the trusted device for the corresponding request message (method M6) and stored on it until the response message is received from the crypto device. The benefit is analogous to the previous method that no key verification need be done by the trusted device. A combination of this method with the previous method is useful.

### 4.4.5. Methods for Mobility

#### M10. Storage of Personal Data on Mobile Device

*Special Requirement on Device:*
- ability to carry the device easily on one's person

The benefit of this method is that the customer is mobile. Furthermore the SET software on the workstation need not be personalized. Only the implementation of a general SET customer software is needed. All personal data is stored on the customer's mobile device. If the mobile device does not have enough memory capacity to store the signature key certificate, the certificate can also be requested by the workstation from the Internet. As listed in paragraph 4.3.2 the personal data is the following.

| *Personal Data* |
|---|
| *Cryptographic Data for Identification*<br>• Private Signature Key<br><br>*Account Data*<br>• Private Account Number (PAN);<br>including the Bank Identification Number (BIN)<br>• Date of Card Expiry (CardExpiry)<br>• Secret of Private Account Number (PANSecret)<br><br>*Further Personal Secret Data*<br>• CardSecret<br>(for Transaction Validation by non-SET system, see [MaVi97b, p.56])<br><br>*Further Personal But Not Secret Data*<br>• (Signature Key Certificate)<br>• Identification of Payment Card Brand (BrandID) |

For every purchase the personal data must be loaded from the mobile device to the workstation. Hence the SET customer software is personalized directly before a purchase. After a transaction the personal data as well as the recorded transaction data should be deleted from the workstation. This method can easily be combined that methods which prevent the theft of particular personal data by performing certain processes on the device. Thus the SET customer system is split into a general part and a personalized part.

#### M11. Storage of Transaction Data

*Special Requirement on Device:*
- possibly extra storage device

During a transaction the SET customer software creates a transaction database [MaVi97a, p.62]. All data needed to process the transaction must be stored in this database. Examples are RRPIDs to be able to relate a received response to its corresponding request [MaVi97b, p.295] or the challenge values (Chall-C, Chall-C2) to be able to check the freshness of a response messages [MaVi97b, p.308]. Furthermore in current SET implementations the whole messages sent or received by the customer software are stored.

If a mobile device is used together with general (not personalized) SET customer software on the workstation (method M10), the transaction database is generated on the workstation and later not accessible to the customer. There are two reasons why particular data should be stored in this case by other means: for general information of the customer about his effected purchases and for dispute handling. General information about his purchases could be given to the customer by a few essential elements stored on the device (method M3).

For dispute handling only the received messages, which are signed, and the non-transmitted elements of HODInput are important. Using the request messages, the customer can prove nothing because request messages with any content can be generated by the customer at any time. The elements of HODInput can easily be stored on a trusted device (possibly already with method M3), because this structure is only 36 bytes long plus the length of the order description (see appendix 2). For example the elements of HODInput are needed for disputes in which the merchant claims that he has lost the exact order description (OD) of an effected purchase request and disagrees with the customer about the description of this order.

Because the received messages are signed as whole messages, the whole messages must be stored. If the mobile device has insufficient storage capacity to store this data, an alternative is a remote stationary storage device and a procedure similar to the procedure described in method M8. With this procedure copies of the response messages are forwarded to the remote device for later use. This is not described in more detail, because it should first be examined what it is provable with the data stored by the customer (which is not done in this work) and because this data is not needed for some dispute handlings as described below.

The transaction information must be stored on merchant and payment gateway systems for a minimum time period specified by the payment brand [MaVi97b, p.5]. Therefore at least certain disputes can be handled for this period without data stored by the customer.

## 4.5. Implementation Proposals

In the following, implementation proposals are given for two scenarios considered to become the most typical in the next future.

### 4.5.1. Proposal I (Only Smartcard Trusted)

*Scenario:*

The customer's SET system consists of a workstation not trusted by the customer and a (trusted) smartcard. A trusted user input and output facility is not provided. This is the case if the smartcard reader (which is regarded as part of the workstation) has no display and keypad or the display and keypad is not trusted by the customer. This scenario is considered to become typical for the near future, because cheap and simple smartcard readers that cannot be trusted or smartcard readers without display and keypad (e.g. a reader included in a floppy disk) will presumably become common.

Furthermore in this scenario it is assumed that at any time before and after a purchase a device is available with a trusted user input and output facility for the predefinition of constants and limits and for to check logged data. This can be an unlinked output device as described for method M3.

*Implementation Proposal:*

This implementation proposal is a combination of methods for signature generation and mobility as listed below. Methods for encryption and signature verification are not included, because the required trusted remote device is not available.

For methods M3, M4, M5 and M6, variation f is used (see paragraph 4.4.2). The elements to be logged or checked by the different methods are printed in italics in the table below. They are determined with the help of the tables in paragraph 4.4.3.

*Methods Used*

| |
|---|
| **M1**: Signature Key and RSA Algorithm on Smartcard |
| **M3**: Logging of Data on Smartcard<br>*(HODInput (incl. PurchAmt, OD), MerchantID, PReqDate)* |
| **M4**: Predefinition of Limits on Smartcard<br>*(all limits as defined in M4 except for limits per time)* |
| **M5**: Predefinition of Constants on Smartcard<br>*(Language, SWIdent, Account Data)* |
| **M6**: Generation of Elements on Smartcard<br>(*HOD, TransStain, ODSalt)* |
| **M10**: Storage of Personal Data on Smartcard |

The element Chall-C is not generated with method M6, because it is only useful if methods for signature verification are also implemented. The HODInput structure is logged completely for dispute handling (see method M11). The MerchantID is logged although it is difficult for the customer to link this identifier to a real merchant (for an alternative technique see method M7). For the sake of simplicity it is assumed that the element InstallRecurData (in HODInput) is not used.

For the procedure in figure 4.26 the following assumptions are made.

- Constants and limits are already predefined on the smartcard.
- The PInitReq/Res message pair has already been exchanged.

The workstation need not load personal data from the smartcard (as described for method M10), because the workstation need not populate fields of the input structures which will be populated by the smartcard. The figure 4.26 shows which part of the procedure is to be implemented on the untrusted workstation and which part is to be implemented on the smartcard. On the smartcard the operations must be processed one after the other, which is indicated by arrows. The output contains not only the RSA signature but also the elements ODSalt and TransStain which are generated on the smartcard and which are needed for the further processing on the workstation.

The following table summarizes the functionality required on the smartcard.

| *Needed Functionality* |
|---|
| • Protected Mode for Predefining Limits/Constants<br>• Logging of Elements<br>• Check of Elements with the Help of Predefined Limits<br>• Check of Constant Elements<br>• Random Number Generation (for ODSalt)<br>• HMAC Keyed-Hash Algorithm (for TransStain)<br>• SHA-1 Hash Algorithm<br>• Check of Hashes with Hashes Generated Before<br>• RSA Signature Algorithm<br>• Output of Logged Data (if requested) |

Furthermore the following data, which is not input data, must be stored on the smartcard.

| *Needed Data* |
|---|
| • Predefined Limits<br>• Predefined Constant Values<br>• Random Seed or Pool (for Random Number Generation)<br>• Personal Data (without signature key certificate, cf. method M10) |

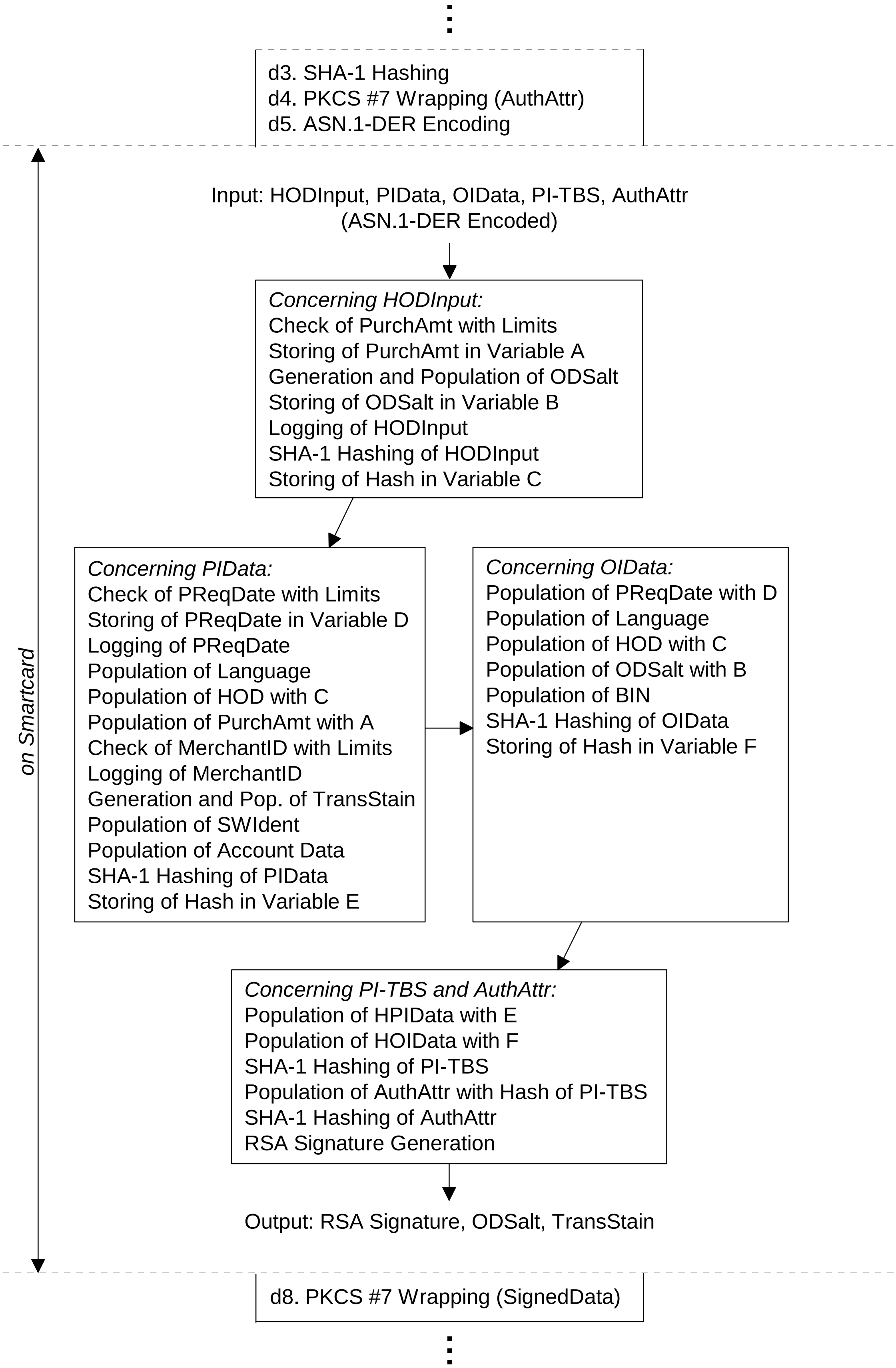


Fig. 4.26: Procedure for Proposal I

### 4.5.2. Proposal II (Smartcard and User I/O Trusted)

*Scenario:*

The customer's SET system consists of a workstation not trusted by the customer and a (trusted) smartcard. Furthermore a user input and output facility trusted by the customer (as described in iii. and iv. of section 3.3) is provided by a smartcard reader. This smartcard reader is a trusted device with a display and a keypad. The workstation and the reader are the general part of the customer system (cf. method M10) and the smartcard is the personalized part. The customer system is split into the untrusted part consisting of the workstation and the trusted part consisting of the reader and smartcard. Because in this scenario the reader is a trusted but not a private device, as described in paragraph 3.1.2, the reader must be tamper-resistant. This requirement on the reader makes it more expensive than in the previous scenario. Nevertheless this scenario is considered to be important for the near future, because it enables the purchase content to be checked personally on a trusted device.

*Implementation Proposal:*

As proposal I this implementation proposal is a combination of methods for signature generation and mobility without methods for encryption and signature verification.

Unlike method M4, method M7 is used to check the element PurchAmt, the merchant's name, the element PReqDate and the element OD. As described in method M7 not the MerchantID is checked but the merchant's name in clear text (hereafter called MName). These elements are checked on the smartcard reader. Therefore the procedure to be implemented on trusted devices is divided into a part for the smartcard reader and another part for the smartcard. This is shown in figure 4.27. All operations on the smartcard that are changed regarding the procedure of proposal I are printed in bold.

The purpose of method M3 for this implementation proposal is not primarily the check of data afterwards as in the previous proposal but the storage of important data for dispute handling (see M11).

*Methods Used*

| |
|---|
| **M1**: Signature Key and RSA Algorithm on Smartcard |
| **M3**: Logging of Data on Smartcard<br>*(HODInput (incl. PurchAmt, OD), MName, PReqDate)* |
| **M5**: Predefinition of Constants on Smartcard<br>*(Language, SWIdent, Account Data)* |
| **M6**: Generation of Elements on Smartcard<br>(*HOD, TransStain, ODSalt)* |
| **M7**: Personal Check of Elements on Smartcard<br>*(PurchAmt, MName, PReqDate, OD)* |
| **M10**: Storage of Personal Data on Smartcard |

As in the previous proposal the element Chall-C is not generated on the smartcard and, for the sake of simplicity, it is assumed that the element InstallRecurData (in HODInput) is not used.

For the procedure in figure 4.27 the following assumptions are made.

- The constants are already predefined on the smartcard.
- The PInitReq/Res message pair has already been exchanged.

The following table summarizes the functionality required on the smartcard.

| *Needed Functionality* |
|---|
| • Protected Mode for Predefining Constants<br>• Logging of Elements<br>• Check of Constant Elements<br>• Random Number Generation (for ODSalt)<br>• HMAC Keyed-Hash Algorithm (for TransStain)<br>• SHA-1 Hash Algorithm<br>• Check of Hashes with Hashes Generated Before<br>• RSA Signature Algorithm<br>• Output of Logged Data (if requested) |

Furthermore the following data, which is not input data, must be stored on the smartcard.

| *Needed Data* |
|---|
| • Predefined Constant Values<br>• Random Seed or Pool (for Random Number Generation)<br>• Personal Data (without signature key certificate, cf. method M10) |

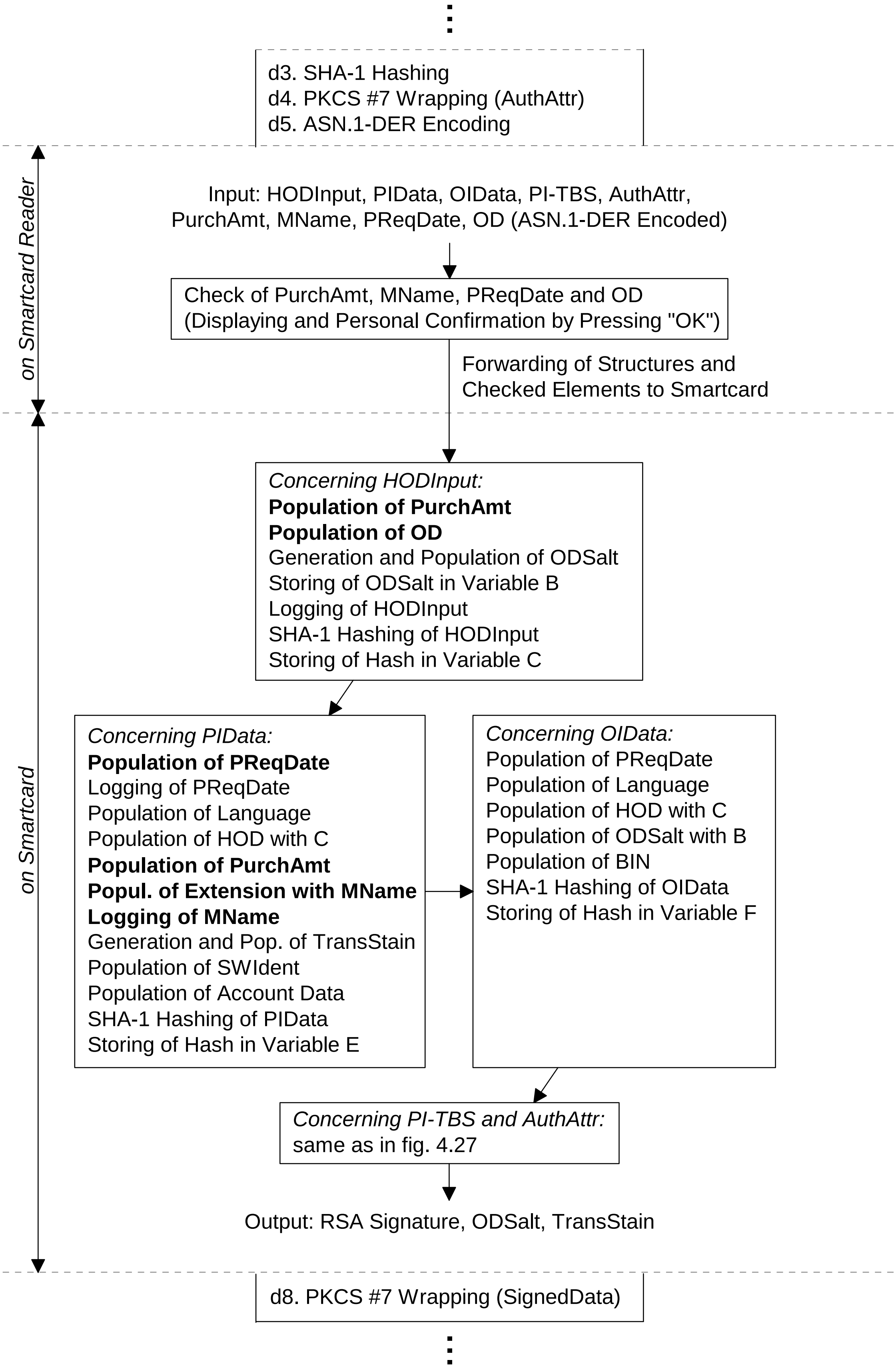


Fig. 4.27: Procedure for Proposal II

# 5. Prospects and Open Problems

In the last chapter implementation proposals are described for two scenarios with smartcards. The main problems of todays smartcards are the lack of sufficient processing power and the need to add an external trusted user interface.

More powerful trusted devices may replace smartcards in the future. For example mobile phones or personal digital assistants (PDA's) have the advantage of more processing power and an included user interface. With this the additional implementation of further security critical parts may be possible or even the implementation of the entire SET customer system. Furthermore with future powerful trusted devices a combination of several payment systems on the same device could be a useful enhancement.

But not only the improvement of trusted devices can achieve a more secure SET customer system. Also changes in the protocol design of SET could solve security problems.

For example the current version of SET uses private account numbers (PAN's) which can also be used for mail or telephone orders (MOTO). New PAN's exclusively used for SET or SET-specific PAN's could prevent fraud with stolen PAN's. Or account data could be omitted at all in the protocol. The customer can clearly be identified with the help of its signature.

Furthermore a consideration of the power limitedness of today's trusted mobile devices in the design phase of the SET customer system and not the implementation phase could avoid problems. A clear separation in the design of tasks to be done by trusted and by untrusted devices could make the implementation easier.

A completely different approach could possibly also solve security problems. In the approach used for this work the assumption is made that the workstation is untrusted. The development of methods to achieve a trusted workstation could make this assumption and additional trusted devices unnecessary.

# Appendix

## Ap.1. Semantics of a Signature

### Ap.1.1. Hand-written Signature

The semantics of a hand-written signature is defined by its context. The term semantics refers to what the author wants to express with his signature and what the consequences of making the signature are. For example a signature for an autograph, a letter, a contract or a cheque have different semantics. The semantics does not have to be expressed explicitly, because it is well-known and defined by its context.

### Ap.1.2. Digital Signature

Concerning the semantics of the digital signature, the following two essential differences from a hand-written signature have to be taken into account.

- The digital signature is not generated by a person but by a device. Based only on the digital signature itself, it can not be seen later whether a person, let alone a specific person, intended to generate the signature. To facilitate that the personal authorization of a specific person can be inferred from a signature, the signature device must have a user input/output facility. Furthermore the signature device and its user input/output facility must be trusted by the specific person (see section 3.3). Therefore one has to distinguish between signatures made on devices with such a user input/output facility and signatures made on devices without such a facility.

- Because the signed data is digital, the semantics of it must be defined explicitly (e.g. that it is ASCII-coded text). Furthermore, if the semantics of the signature is not clear from the text, e.g. if the text is not a letter or a contract but a new type of message, the semantics of the signature must also be defined explicitly.

#### Ap.1.2.1. Pure Digital Signature

A digital signature with no explicit semantics which is made on a device without personal confirmation achieves the following (for description of method *with* personal confirmation see method M2) .

On the conditions that

- the private signature key is accessible exclusively to a definite device *A* and this device makes the key inaccessible to other devices,
- the device has the capability to receive input *m*, to apply the RSA signature function on it and to output the signature *s*,

the signature *s* makes it provable that

- the data *m* was present in the device *A*.

In general, the purpose of a digital signature is to provide a means for an entity to bind its identity to a piece of information. Here, instead of the term entity, the term device is used to make clear that not a person but only a device is meant. The piece of information here is the data *m*, which is normally only a hash.

### Ap.1.2.2. The Digital Signature with Explicit Semantics

*Semantics of m*

The semantics of *m* could be, for example, the statement that the digital data *m* represents ASCII-coded text and that the semantics of the text (in a clearly recognizable human language) is the common semantics of that language.

*Semantics of s*

For some semantics of a signature *s* further requirements (as described in section 3.3) on the signature device are important. In the following, several possible semantics are listed. For each semantics, examples are given of signatures with such semantics.

a. *"Data origin authentication is given."*
   No further semantics is given. The entity is not a person but the device. This corresponds to the case where no explicit semantics is given (cf. Ap.1.2.1).
   Examples: signature with method M1 (sufficient for element LID-C or LID-M).

b. *"This data is checked by this device."*
   The data is either a predefined value on the device or it is generated by the device. If the data cannot be exactly predefined or generated, it is possibly merely compared with predefined limits.
   Examples: signature with method M5 or M6 (sufficient for element Language or PReqDate).

The following is an example of a signature semantics that requires a trusted user control facility (see iii. in section 3.3).

c. *"The signature has been confirmed by the person."*
   The data is neither checked by the device nor by the person but the person intended to make a signature. Hence the value of the signed data is provably defined. If further data is signed with this signature, this data is provably bound to that value.
   Example: signature with method M2 (sufficient for element XID).

The following is an example of a signature semantics that requires a trusted user input/output facility (see iii. and iv. in section 3.3).

d. *"The person checked this data personally."*
   Examples: signature with method M7 (for PurchAmt or MerchantID).

## Ap.2. Length of Data Structures

The memory capacity of a smartcard is limited. Therefore among other things an essential criterion that indicates whether a particular method can be implemented on a particular smartcard is the length of data to be stored on the card during execution time. In the following the length is determined of all data structures used to generate the RSA signature for the message PReq (for the links between these structures cf. fig. 2.6 and fig. 4.24). Thus the required storage memory for logging structures (method M3) is given. Furthermore this allows one to determine whether the input structures for the implementation proposals can be stored entirely and together on the smartcard for processing, or if they must be input one after the other.

Because data structure extensions are not defined in the specifications and their length is not limited, it is assumed here that they are not used. If a length of an element can vary, both the minimum and the maximum are used for further calculations. If for practical implementations a less than maximum value can be assumed, this is given in parentheses. The value "max" means that the length of this element is not limited according to the specification.

| **AuthenticatedAttributes (AuthAttr)** | Length of Payload (in Bytes) | Length of ASN.1 Code (in Bytes) |
|---|---:|---:|
| *ASN.1 Header* | | 2 |
| ContentType | 4 | 21 |
| MessageDigest | 20 | 37 |
| AuthenticatedAttributes | 24 | 60 |

| **DigestedData (DD)** (HPIData, HOIData, HOD) | Length of Payload (in Bytes) | Length of ASN.1 Code (in Bytes) |
|---|---:|---:|
| *ASN.1 Header* | | 2 |
| Version | 1 | 3 |
| DigestAlgorithm | 5 | 11 |
| ContentInfo | 4 | 8 |
| Digest | 20 | 22 |
| DigestedData | 30 | 46 |

| **PIData** | Length of Payload (in Bytes) | Length of ASN.1 Code (in Bytes) |
|---|---:|---:|
| *ASN.1 Header* | | 4 |
| PIHead | 159-486(248) | 210-539(299) |
| PANData | 62 | 72 |
| PIData | 221-548(310) | 286-615(375) |

| **PIHead*** | Length of Payload (in Bytes) | Length of ASN.1 Code (in Bytes) |
|---|---|---|
| *ASN.1 Header* | | 3 |
| TransIDs | 74-114(82) | 86-126(94) |
| Inputs | 34-37 | 60-63 |
| MerchantID | 1-30 | 3-32 |
| TransStain | 20 | 22 |
| SWIdent | 1-256(50) | 3-260(52) |
| AcqBackKeyData | 29 | 33 |
| PIHead | 159-486(248) | 210-539(299) |

*assumption: InstallRecurData and PIExtensions not used

| **TransIDs*** | Length of Payload (in Bytes) | Length of ASN.1 Code (in Bytes) |
|---|---|---|
| *ASN.1 Header* | | 2 |
| LID-C | 20 | 22 |
| LID-M** | 20 | 22 |
| XID | 20 | 22 |
| PReqDate | 13-19 | 15-21 |
| Language | 1-35(3) | 3-37(5) |
| TransIDs | 74-114(82) | 86-126(94) |

*assumption: PaySysID not used

**assumption: PInit messages have been exchanged and therefore LID-M is included in TransIDs.

| **Inputs** | Length of Payload (in Bytes) | Length of ASN.1 Code (in Bytes) |
|---|---|---|
| *ASN.1 Header* | | 2 |
| HOD | 30 | 46 |
| PurchAmt | 4-7 | 12-15 |
| Inputs | 34-37 | 60-63 |

| **PurchAmt** | Length of Payload (in Bytes) | Length of ASN.1 Code (in Bytes) |
|---|---|---|
| *ASN.1 Header* | | 2 |
| Currency | 2 | 4 |
| Amount | 1-4 | 3-6 |
| AmtExp10 | 1 | 3 |
| PurchAmt | 4-7 | 12-15 |

| **OIData*** | Length of Payload (in Bytes) | Length of ASN.1 Code (in Bytes) |
|---|---|---|
| *ASN.1 Header* | | 3 |
| TransIDs | 74-114(82) | 86-126(94) |
| RRPID | 20 | 22 |
| Chall-C | 20 | 22 |
| HOD | 30 | 46 |
| ODSalt | 20 | 22 |
| Chall-M** | 20 | 22 |
| OIData | 184-224(192) | 223-263(231) |

*assumption: extensions not used.
**assumption: PInit messages have been exchanged and therefore Chall-M is included in OIData and BrandID not.

| **HODInput** | Length of Payload (in Bytes) | Length of ASN.1 Code (in Bytes) |
|---|---|---|
| *ASN.1 Header* | | 2-4 |
| OD | 1-max | 3-max |
| PurchAmt | 4-7 | 12-15 |
| ODSalt | 20 | 22 |
| HODInput | 25-max | 39-max |

*assumption: InstallRecurData and ODExtensions not used

# References

## SET

BGHH95 *Mihir Bellare, Juan A. Garay, Ralf Hauser, Amir Herzberg, Hugo Krawczyk, Michael Steiner, Gene Tsudik, Michael Waidner:*
**iKP - A Family of Secure Electronic Payment Protocols;**
USENIX, Jul.1995, pp. 89-106.
<http://www.zurich.ibm.com/Technology/Security/extern/ecommerce/iKP.html>

MaVi97a *Mastercard, Visa:*
**SET Secure Electronic Transactions Protocol,**
**Book One: Business Specifications;**
Version 1.0, May 1997, 72 pages.
<http://www.setco.org/download.html>

MaVi97b *Mastercard, Visa:*
**SET Secure Electronic Transactions Protocol,**
**Book Two: Technical Specification;**
Version 1.0, May 1997, 619 pages.
<http://www.setco.org/download.html>

MaVi97c *Mastercard, Visa:*
**SET Secure Electronic Transactions Protocol,**
**Book 3: Formal Protocol Definition;**
Version 1.0, May 1997, 254 pages.
<http://www.setco.org/download.html>

Schw97 *Thomas Schweinberger:*
**Entwurf eines einheitlichen auf Java basierenden Konzeptes zur Entwicklung von Smartcard Applikationen;**
Diplomarbeit am Wilhelm-Schickard-Institut für Informatik,
Eberhard-Karls-Universität Tübingen, Jun. 1997, pp. 50-68.

## TRUSTED DEVICES AND SMARTCARDS


AnKu96 *Ross J. Anderson, Markus Kuhn:*
**Tamper Resistance - A Cautionary Note;**
Second USENIX Workshop on Electronic Commerce, Oakland, California, USENIX, Nov. 1996, pp. 1-11.

Hend97 *Mike Hendry:*
**Smart Card Security and Applications;**
ISBN 0890069530,
Artech House, Sep. 1997, 210 pages.

Meye97 *Carsten Meyer:*
**Gold und Plastik;**
c't Report Geld Online,
Heise-Verlag, Hannover, Mar. 1997, pp. 130-132.

PPSW96 *Andreas Pfitzmann, Birgit Pfitzmann, Matthias Schunter, Michael Waidner*:
**Mobile User Devices and Security Modules: Design for Trustworthiness**;
IBM Research Report RZ 2784 (#89262) 02/05/96,
IBM Research, Zurich, Feb. 1996.
<http://www.semper.org/sirene/publ/PPSW_96MUDandSM.IBM.ps.gz>

ScKo96 *Stefan Schütt, Bernd Kohlgraf:*
**Chipkarten: Technische Merkmale, Normung, Einsatzgebiete;**
ISBN 3-486-23738-1,
Oldenbourg-Verlag, München, 1996, 224 pages.

VoVo96 *Francesco P. Volpe, Safinaz Volpe:*
**Chipkarten: Grundlagen, Technik, Anwendungen;**
ISBN 3-88229-065-X,
Heise-Verlag, Hannover, 1996, 157 pages.

ZoOt94 *Jose Luis Zoreda, Jose Manuel Oton:*
**Smart Cards;**
ISBN 0890066876,
Artech House, Jan. 1994, 244 pages.

## GENERAL


AASW96 *J.L. Abad-Peiro, N. Asokan, Michael Steiner, Michael Waidner:*
**Designing a Generic Payment Service;**
IBM Systems Journal 37(1), Jan. 1998, pp. 72-88.
<http://www.almaden.ibm.com/journal/sj/371/abadpeiro.html>

AJSW97 *N. Asokan, Phillipe A. Janson, Michael Steiner, Michael Waidner:*
**State of the Art in Electronic Payment Systems;**
IEEE Computer 30(9), Sep. 1997, pp. 28-35.

Mene96 *Alfred J. Menezes, Paul C. van Oorschot, Scott A. Vanstone:*
**Handbook of Applied Cryptography;**
ISBN 0849385237,
CRC Press, Oct. 1996, 816 pages.

PfWa96 *Birgit Pfitzmann, Michael Waidner:*
**Properties of Payment Systems: General Definition Sketch and Classification;**
IBM Research Report RZ 2823 (#90126),
IBM Research, May 1996, 29 pages.

Schn94 *Bruce Schneier:*
**Applied Cryptography;**
ISBN 0471597562,
John Wiley & Sons, Jan. 1994, 618 pages.