

# Non-Repudiation in SET: Open Issues

Els Van Herreweghen

IBM Research, Zurich Research Laboratory, CH-8803 Rüschlikon, Switzerland  
evh@zurich.ibm.com

**Abstract.** The SET payment protocol uses digital signatures to authenticate messages and authorize transactions. It is assumed that these digital signatures make authorizations non-repudiable, i.e., provable to a third-party verifier. This paper evaluates what can be proved with the digital signatures in SET. The analysis shows that even a successful and completed SET protocol run does not give the parties enough evidence to prove certain important transaction features. A comparison with the similarly-structured *iKP* protocol shows a number of advantages of *iKP* as opposed to SET with respect to the use of its signatures as evidence tokens. It is shown that non-repudiation requires more than digitally signing authorization messages. Most importantly, protocols claiming non-repudiation should explicitly specify the rules to be used for deriving authorization statements from digitally signed messages.

## 1 Introduction

Digital signatures in the *iKP* [BGH<sup>+</sup>95, BGH<sup>+</sup>00] payment protocol are intended by design to represent non-repudiable message receipts for customer, merchant and bank. The SET [MV97] protocols are similar in design to *iKP*; though the SET specifications do not claim non-repudiability of the SET signatures, press releases and public opinion tend to attach this feature to them. In this paper, we investigate the value of SET digital signatures as evidence towards an external verifier. A verifier may be a subsystem of a third-party dispute handler that arbitrates payment disputes, or an online ombuds service as in [Kat96]. Alternatively, it can be a stand-alone system used by a company's accounting department or a national tax department to verify digital receipts submitted as evidence of transactions. The latter use illustrates the need for digital receipts to be self-contained; they should be usable as valid transaction receipts, without the need for additional evidence collected from semi-trusted parties such as banks.

A fair amount of work has been done recently on formal specification and verification of e-commerce protocols [Kai95, KN98, Bol97, MS98, Bra97]. Both [Kai95] and [KN98] focus on accountability by introducing authentication logics modified with predicates of the form "A can prove X to B". The statements X in both logics are payment system dependent; [KN98] applies this logic to SET to prove the following: a Merchant M, having received a PReq (Payment Request) message from Customer C with Order Instructions OI, can derive that *M believes M can-prove (C said OI) to J*, where J is an independent verifier.

In this paper, we define a number of concrete, protocol-independent assertions that parties participating in a payment protocol may want to prove; we evaluate the evidence collected by SET as to its usability in proving those statements; and compare SET with *i*KP. We do not prove the protocols to be secure, in the sense of resistance against outsider and replay attacks. Rather, we focus on the provability of the respective authorizations to an external verifier.

In Sect. 2, we introduce a high-level protocol, representing protocols such as SET or *i*KP. In Sect. 3, we describe what kind of statements each participant in this payment protocol may want to prove, using the evidence collected during a payment run. The statements form a subset of the generic payment claim language described in [AHS98]. In Sect. 4, we describe the actual SET protocol, and evaluate its evidence tokens against the requirements described in Sect. 3. We show that some authorizations cannot be proved, even with evidence collected from a correct SET transaction. For example, a customer has no secure receipt after a successfully completed SET protocol run. Section 4 then concludes with a number of recommendations for constructing provable payment protocols. Section 5 illustrates how *i*KP differs from SET with respect to some of the provability requirements. First, the evidence tokens in *i*KP are more powerful (more authorizations can be proved). Second, *i*KP, more so than SET, is designed with the principal of minimal information disclosure in mind: individual payment parameters (such as the payment amount) can be proved without having to reveal other data (such as a description of goods purchased). Section 6 summarizes our findings. Secure authorization using digital signatures does not guarantee that the authorization and its parameters can be proved. Therefore, it is recommended that protocols claiming non-repudiation explicitly specify the rules for deriving authorization statements from digitally signed messages.

## 2 High-Level Payment Protocol Description

The SET and *i*KP protocols can both be represented by the high-level protocol depicted in Fig. 1. Before the start of the actual payment protocol, Customer and Merchant have agreed on a description and price of the goods. In an optional Initiate - Invoice exchange, Customer and Merchant exchange variables, options and randomizers needed in the ensuing payment messages. The Customer then sends a Pay-Request message to the Merchant, which the Merchant uses to produce an Auth-Request asking authorization from the Acquirer. The Acquirer goes through the financial network to obtain payment authorization and returns an Auth-Response to the Merchant, indicating success or failure, and optionally the actually captured amount. The Merchant then produces a Pay-Response (Confirm) message and sends it to the Customer.

SET, as well as *i*KP, represents different protocol versions depending on which parties (all, only Merchant and Acquirer, only Acquirer) digitally sign messages. This paper analyzes the evidence collected in the most secure versions, i.e. the protocol variants where all parties digitally sign messages.

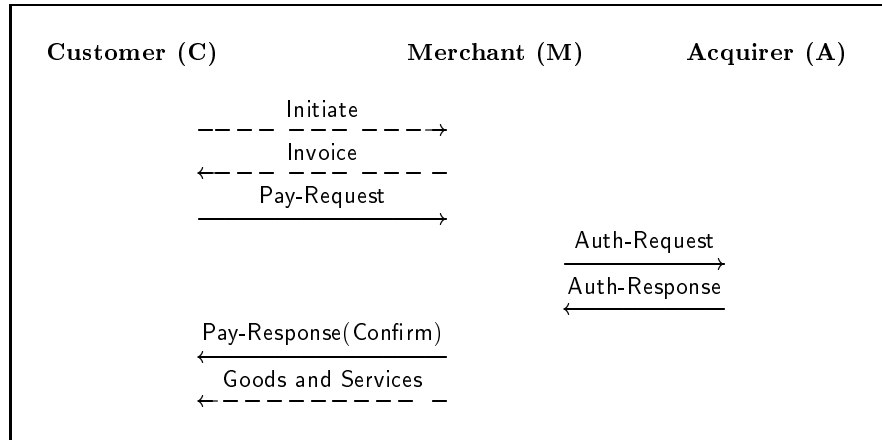


Fig. 1. Generic credit-card payment protocol.

A protocol option set by Merchant (optionally also Customer) determines whether the Auth-Request - Auth-Response exchange results in the actual transfer of money (simultaneous authorization and capture) or whether an ensuing Cap-Request - Cap-Response exchange between Merchant and Acquirer completes the actual transaction (separate authorization and capture). To simplify the analysis, we will assume the former scenario, i.e. simultaneous authorization and capture. For the same reason, we will not consider reversal or refund of payments.

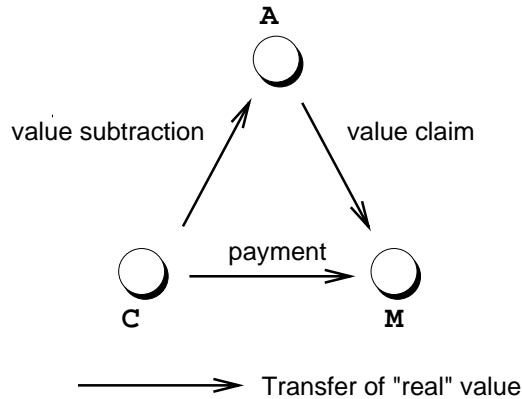
### 3 Proving Authorizations of *primitive transactions*

In [AHS98], a language was presented to express claims (or provable statements) that participants in a payment protocol may want to be able to prove. This language is derived from the generic payment service definition in [APASW98] and is independent of the payment model (stored-value, account-based). A payment transaction involves Customer (C), Merchant (M) and Acquirer (A) and consists of three types of *primitive transactions*, each representing the view of a subset of players on a payment transaction:

- In *value subtraction*, C allows A<sup>1</sup> to remove “real value” from C’s account; this implies C’s right to spend “electronic value.”
- In *value claim*, M requests that A deposit “real value” in M’s account.
- In *payment*, C transfers value to M.

In stored-value (electronic purse) systems, *value subtraction*, *payment* and *value claim* may be separate transactions, taking place at different points in

<sup>1</sup> actually: C’s own issuing bank through A.



**Fig. 2.** Value transfer transactions.

time. In the account-based protocol in Fig. 1, where the Acquirer is involved online and authorization and capture occur simultaneously, *value subtraction*, *payment* and *value claim* finish at the same time and cannot be separated. Distinguishing among the three *primitive transactions*, however, enables us to formulate disputes between each subset of players: it allows us to reason about the bank's behavior as well, as it receives and sends authorization messages to the other parties.

After completion of the payment protocol in Fig. 1, each of the parties (C, M and A) ideally has non-repudiable proof of transaction authorization by the other two parties. Using a notation similar to the one used in [AHS98], the following requirements are then fulfilled:

- M can prove “C **authorized payment** (C, M, Amount, Date, Ref)”.  
Ref stands for a set of reference parameters, which may be protocol-specific, such as transaction identifiers or description of the goods purchased. We will further abbreviate the statement “C **authorized payment** (...)” to “Auth(C-M)”.
- C can prove “M **authorized payment** (C, M, Amount, Date, Ref)”, or: “Auth(M-C)”.
- A can prove “C **authorized value subtraction** (C, A, Amount, Date, Ref)”, or: “Auth(C-A)”.
- C can prove “A **authorized value subtraction** (C, A, Amount, Date, Ref)”, or: “Auth(A-C)”.
- A can prove “M **authorized value claim** (M, A, Amount, Date, Ref)”, or: “Auth(M-A)”.
- M can prove “A **authorized value claim** (M, A, Amount, Date, Ref)”, or: Auth(A-M)”.

For a protocol following the scheme in Fig. 1, Pay-Request must then contain proof of Auth(C-M) and Auth(C-A); Auth-Request contains proof of Auth(M-A)

and forwards proof of  $\text{Auth}(C-A)$ ;  $\text{Auth-Response}$  contains proof of  $\text{Auth}(A-M)$  and  $\text{Auth}(A-C)$ ;  $\text{Pay-Response}$ , finally, contains proof of  $\text{Auth}(M-C)$  and forwards proof of  $\text{Auth}(A-C)$ . Thus, after receiving  $\text{Pay-Request}$ , M can prove  $\text{Auth}(C-M)$ ; after receiving  $\text{Auth-Request}$ , A can prove  $\text{Auth}(C-A)$  and  $\text{Auth}(M-A)$ ; after receiving  $\text{Auth-Response}$ , M can prove  $\text{Auth}(A-M)$  (in addition to  $\text{Auth}(C-M)$ ); and after receiving  $\text{Pay-Response}$ , C can prove  $\text{Auth}(A-C)$  and  $\text{Auth}(M-C)$ .

If all of the above requirements are fulfilled, all the parties in a completed transaction have sufficient proof to convince a verifier of their view of the primitive transactions in which they were involved.

The absence of one of these provable authorizations need not make the payment protocol insecure, but it may make disputes between the parties involved in the specific *primitive transaction* more difficult (e.g., having to rely on evidence provided by other parties) or even impossible without resorting to off-line means (e.g., an account statement proving a debit and thereby a payment). In the following, we highlight the importance of the respective authorizations:

- $\text{Auth}(M-C)$  is equivalent to a receipt of payment by M. In its absence, C cannot prove the payment to M without involving evidence collected by A, or without resorting to off-line means (e.g., account statement).
- $\text{Auth}(A-C)$  is equivalent to a receipt by A of the amount deducted from C's account and can potentially be replaced by an off-line account statement. It allows C to verify that the amount deducted by A and the amount in M's receipt are equal.
- $\text{Auth}(C-M)$  is obviously important if M delays the actual authorization/capture and makes certain decisions (such as shipping goods, or reserving stock) based on  $\text{Auth}(C-M)$  before contacting A. This authorization may seem less important in the case where M directly captures the money and therefore does not need  $\text{Auth}(C-M)$  as a payment guarantee. However,  $\text{Auth}(C-M)$  still has its value as a proof of terms and conditions of the payment, such as description of the goods (which, in case of disagreement, M may otherwise not be able to prove).
- $\text{Auth}(C-A)$  is A's proof that C entitled A (or its own issuing bank through A) to deduct money from C's account.
- $\text{Auth}(M-A)$  is A's proof that M asked the payment to be made to him.
- $\text{Auth}(A-M)$  is M's proof that A transferred (or committed to transfer) the money to M's account.

When analyzing SET in Sect. 4, we will analyze for each of the above authorizations whether the interested party has the evidence needed to prove it, whether it needs other parties to cooperate in giving evidence, or whether the statement cannot be proved at all. In SET, some of the authorizations cannot be proved, or can only be proved with the help of evidence collected by other parties. *iKP*, as discussed in Sect. 5, gives each party enough evidence to prove the other parties' authorizations.

## 4 SET

### 4.1 SET Protocol Overview

The basic SET protocol follows the model of Fig. 1. For reasons of simplicity, we do not consider additional SET options, such as separation of authorization and clearing, refunds or credit, or the splitting of a single payment transaction into multiple authorization threads.

A number of SET messages are encrypted using the public key of the recipient. Depending on the security requirements of the message, and depending on whether certain parts of the message are already in encrypted form, different types of encryption are used. From an evidence point of view, however, these different types of encryption are equivalent because the actual evidence to an external verifier necessarily consists of the decrypted message. Therefore, we simplify the encryption notation:  $E_X(M)$  stands for a message encrypted using X's public encryption key  $PK_X$  from which X can retrieve M by decryption. In most cases, M (or a part of M) is encrypted using a symmetric (DES) key, which is (part of) the actual data encrypted with  $PK_X$ .

For further simplification, we omit a number of fields that are optional or not relevant to verifiability, such as thumbprints, bank identification numbers, language identifiers, request-response pair identifiers (RRPIDs), and optional extensions. Figure 3 represents the atomic and composite fields used in SET; Fig. 4 shows the protocol messages.

In the following sections, we describe which pieces of evidence can prove the statements described in Sect. 3. We do so by formulating "rules" that a verifier V, when presented with pieces of SET evidence, can use to infer conclusions about those statements. A rule of the form

<b>evidence</b> (Evidence1, Evidence2, ...), <b>verify</b> (Statement1, Statement2, ...) → <b>conclusion</b>
--

expresses that, if V is given the pieces of evidence in **evidence**, and if V can also verify statements (equalities, cryptographic relationships) in **verify**, then V will arrive at **conclusion** (or, equivalently: *then conclusion* can be proved to V). The rules used are not rules in any logic; rather, we use them as a convenient intuitive notation for deriving assertions from signed messages.

In the following analysis, we assume an implicit linking between keys and entities: if message M is signed with a secret key  $K_s$  of which the public counterpart  $K_p$  is certified by a trusted Certification Authority as belonging to entity E, then M is signed by E. Here we also assume an implicit distribution and verification of the certificates certifying and carrying the  $K_p$ 's; in the following, when V can convince itself that P signed message M at time T, V was presented with a certificate for P's public key  $K_p$ , which was valid at time T.

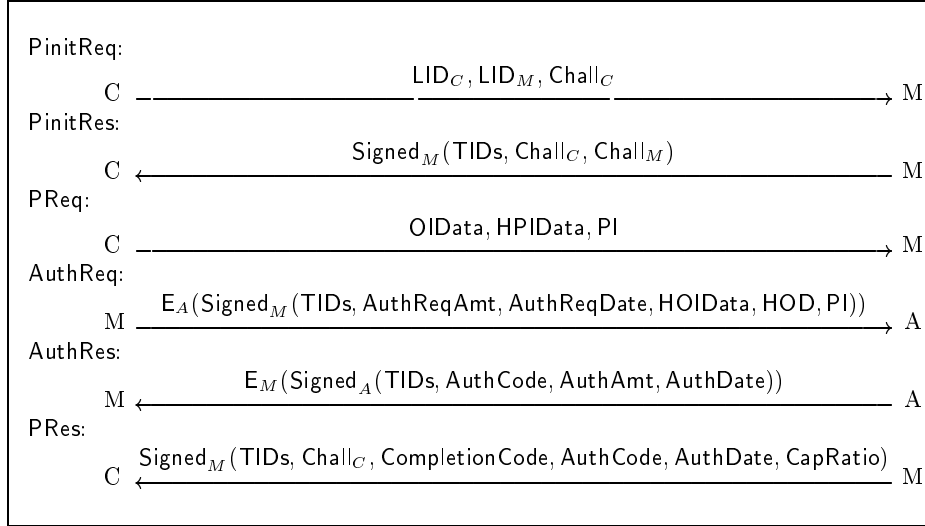
H(M)	image of M under a strong collision-resistant one-way hash function
$E_X(M)$	message M, encrypted under X's public encryption key
$\text{Signed}_X(M)$	message M, signed with X's private signature key (includes M)
$S_X(M)$	signature-only of M with X's private signature key (does not include M)
$E_X(\text{Signed}_Y(M, \text{baggage}))$	simplified notation for $\{\text{Signed}_Y(M, \text{baggage}), E_X(M), \text{baggage}\}$ ; baggage is already encrypted and is signed together with encrypted M
TIDs	$LID_C, LID_M, XID, PReqDate, Language$
$LID_Z$	local ID of Z
$Chall_Z$	challenge of Z
XID	globally unique ID
HOD	$H(\text{HODContents})$
HODContents	OD, PurchAmt, ODSalt
OD	Order Description
MID	Merchant ID
AuthX, X=Code,Amt,Date	code, amount, date in AuthReq
AuthResX, X=Amt, Date	amount, date in AuthRes
CompletionCode	Merchant's completion code
CapRatio	ratio of AuthAmt:PurchAmt
PANData	PAN (Private Account Number), CardExpiry, PANSecret, EXNonce
PI	$S_C(\text{HOIData}, \text{HPIData}), E_A(\text{PIHead}, \text{HOIData}, \text{PANData})$
HOIData	$H(\text{OIData})$
HPIData	$H(\text{PIData})$
OIData	TIDs, $Chall_C$ , HOD, ODSalt, $Chall_M$
PIData	PIHead, PANData
PIHead	TIDs, HOD, PurchAmt, MID

Fig. 3. SET: Atomic and composite fields.

## 4.2 Proofs of Authorizations

**M proving Auth(C-M).** After receiving PReq, M can use the non-encrypted part (OIData, PI, HPIData) to prove Auth(C-M).<sup>2</sup> As PurchAmt is only present in OIData in hashed (HOD) form, V also needs HODContents to verify the PurchAmt in PReq. M thus has to reveal OD to V in order to prove Auth(C-M) for a certain PurchAmt.

<sup>2</sup> *composite-field[component-fields]* denotes a composite field (e.g., OIData) with its individual components.



**Fig. 4.** SET: Payment with in-line authorization.

The verifications (**verify**) performed on the evidence convince V that C signed a valid hash HOIData over the elements in OIData, including HOD; by verifying that HOD is a valid hash over HODContents including PurchAmt and OD, V concludes that C signed these data as well, and thus that C authorized the payment. Ref() is an unordered list of additional reference data about the payment.

```

evidence (OIData [TIDs, ChallC, HOD, ODSalt, ChallM], HPIData, PI [SigC, ...],
          HODContents [OD, PurchAmt, ODSalt]),
verify (HOIDATA = H(OIData),
        HOD = H(OD, PurchAmt, ODSalt),
        SigC = SC(HOIDATA, HPIData))
→ C authorized
    payment (C, M=?, PurchAmt, TIDs.PReqDate, Ref(TIDs, HOD, OD)).

```

Note that, with PReq and HODContents together, M cannot prove the payment was made to him. Only A can prove that by revealing the decrypted and signed PI (including MID). However, PReq and HODContents can still be used as proof of terms and conditions of the payment agreed to by C, as discussed in Sect. 3.

**C proving Auth(M-C).** C can try to use the receipt PRes to prove Auth(M-C). PRes includes a CapRatio (ratio of captured amount CapAmt to authorized

amount  $PurchAmt$ ), but does not include  $HOD$ ,  $PurchAmt$  or  $C$ 's identity. Thus,  $PRes$  alone cannot be used as a receipt for a certain  $CapAmt$ .

<p><b>evidence</b> <math>^a(\text{Signed}_M(\text{TIDs}, \text{Chall}_C, \text{CompletionCode}, \text{AuthCode}, \text{AuthDate}, \text{CapRatio}))</math>  <math>\rightarrow M</math> <b>authorized payment</b> (<math>C=?</math>, <math>M</math>, <math>Amt=?</math>, <math>AuthDate</math>, <math>Ref(\text{TIDs}, \text{CapRatio})</math>).</p>
---

<p><math>^a</math> <b>evidence</b> (<math>\text{Signed}_X(M)</math>) is used as a short form for:  <b>evidence</b> (<math>\text{Sig}_X, M</math>), <b>verify</b> (<math>\text{Sig}_X = S_X(M)</math>).</p>
--

What if  $C$  additionally provides  $PReq$  and  $HODContents$  to  $V$ , proving  $C$ 's identity and  $PurchAmt$  associated with this  $TIDs$ ? Unfortunately, this does not preclude a dishonest  $C$  from making up a  $PReq$  on the spot containing  $PurchAmt$  and identity  $C$  wants to prove. Or, from a  $PReq$  proving

$C$  **authorized payment** ( $C, M, PurchAmt, AuthDate, Ref(\text{TIDs})$ )

and a  $PRes$  proving

$M$  **authorized payment** ( $C, M, Amt=?$ ,  $AuthDate, Ref(\text{TIDs}, \text{CapRatio})$ ),

$V$  cannot conclude

$M$  **authorized payment** ( $C, M, PurchAmt * CapRatio, AuthDate, Ref(\text{TIDs})$ )

unless  $C$  can prove to  $V$  that it was indeed that specific  $PReq$  ( $C$  and  $PurchAmt$ )  $M$  replied to. In [Van99], additional rules are formulated taking extra evidence from  $M$  or  $A$  into account to convince  $V$  of the  $OIData$  agreed to by  $C$  and  $M$  in the actual transaction. However,  $C$ , with only the evidence collected in a correct protocol run, has no secure receipt of the transaction. Practical consequences are described in more detail in Sect. 4.3.

**A proving Auth(C-A).** To prove authorization by  $C$ ,  $A$  can use  $S_C(\text{HOIData}, \text{HPIData})$  in  $AuthReq$ . Because  $A$ , without explicit cooperation from  $M$ , does not possess  $OIData$ ,  $A$  tries to prove that  $\text{HPIData} = H(\text{PIData})$ , with  $\text{PIData}$  mapping the authorization data to be proved. This assumes that  $\text{PIData}$  and the cleartext  $\text{PANData}$  are revealed to the verifier:

<p><b>evidence</b> (<math>\text{Sig}_C, \text{PIHead} [\text{TIDs}, \text{HOD}, \text{PurchAmt}, \text{MID}]</math>, <math>\text{HOIData}</math>,  <math>\text{PANData} [\text{PAN}, \text{CardExpiry}, \text{PANSecret}, \text{EXNonce}]</math>),  <b>verify</b> (<math>\text{HPIDATA} = H(\text{PIHead}, \text{PANData})</math>,  <math>\text{Sig}_C = S_C(\text{HOIData}, \text{HPIData})</math>)  <math>\rightarrow C</math> <b>authorized</b>  <i>value subtraction</i> (<math>C, A=?</math>, <math>PurchAmt, \text{TIDs.PReqDate}, \text{Ref}(\text{TIDs}, \text{HOD})</math>).</p>
---

Revealing  $\text{PANData}$  can only be avoided if  $A$  is able to obtain  $OIData$  and  $HODContents$  (e.g., from  $M$ , see [Van99]). However, with the evidence collected by  $A$  alone,  $A$  has to reveal  $\text{PANData}$  to  $V$  in order to prove  $Auth(C-A)$ .

**C proving Auth(A-C).** C never receives proof of A's authorization. C can prove *value subtraction* only when Acquirer or Merchant cooperate and the verifier accepts an Auth-Response (targeted at M) as evidence of *value subtraction* (see [Van99]). But, with the evidence collected by C alone, C cannot prove Auth(A-C).

**A proving Auth(M-A) and M proving Auth(A-M).** These respective authorizations can be derived in a straightforward way from the AuthReq, respectively AuthRes messages, as shown in [Van99].

### 4.3 Discussion

**Problems Related to Missing Authorizations.** M, after receiving a valid PReq, cannot prove the payment was made to him, and has no guarantee that PI contains the correct MID (M's identifier). It can reasonably be argued that this is of minor importance in the assumed on-line authorization scenario because M immediately gets a payment guarantee from A. It is important to see, however, that delayed processing of PReqs would be problematic.

A more severe problem is this: the Customer has secure receipts from neither Merchant (Auth(M-C)) nor Acquirer (Auth(A-C)). One may argue that PRes, in combination with PReq, may be treated by V as a valid receipt, and that cheating by C (by providing a false PReq) is likely to be detected. However, it remains an important point that, in the absence of other evidence provided by M or A, a Customer providing PRes in combination with a false PReq may cheat a verifier.

The consequences of these missing authorizations depend on the default behavior of a verifier presented by C with a (PReq, PRes) pair as proof of a payment. Consider a company optimizing its expense account processing by accepting electronic (SET) receipts from its employees for business-related expenses. The company's accounting department can take three different approaches to verifying these receipts:

- As current SET implementations do not come with the necessary mechanisms to verify transaction records after the fact, the company simply trusts the employee to claim the correct amount, and ignores the actual SET transaction record. This obviously allows errors and fraudulent claims.
- The accounting department decides to invest in verification software that extracts SET receipts (in this case, PRes and PReq) from the employees' transaction records and verifies them. Still, this leaves opportunities for fraud by employees, who are able to tamper with their SET software and make it produce different PReq's for the same TIDs. Suppose an employee buys an airline ticket for US\$200 (PReq with PurchAmt = 200, PRes with CapRatio = 1). The employee then produces another PReq for the same TIDs, but with PurchAmt = 1000. With the second PReq and the PRes, the employee now proves to the accounting department's verifier that she bought a ticket for US\$1000.

- The accounting department can, for every claim, ask for additional statements or SET evidence from the bank. But this approach is costly and certainly does not fulfill the original goal of optimization.

The example above shows how a dishonest Customer can cheat a verifier, if the verifier accepts incomplete proof. The lacking proof of authorization can of course work in C's disadvantage as well. Consider a Customer who pays US\$100 for a certain item described in OD, and considers the receipt of PRes as a valid proof of payment for this item. If the goods delivered do not match the negotiated terms (OD), or C's account is debited with a different amount than expected, C has to rely on additional evidence provided by A or M. This may be a costly procedure; or, it may turn out to be impossible if M or A's payment systems do not support the dispute handling mechanisms required for collecting this evidence.

**Revealing Data.** As the goods description OD is hashed together with PurchAmt in HOD, neither C nor M can prove authorization of *payment* for a specific PurchAmt without revealing OD to V.

More data than necessary is also revealed when A proves Auth(C-A): to prove C's authorization of *value subtraction* with only the evidence in its possession, A needs to reveal the entire contents of PIData, including PANData, to V.

This could be avoided by an additional level of hashing, such that only a hash  $H(\text{PANData})$  is revealed together with PIHead in order for V to recompose HPIData (and thus verify PI without needing PANData).

#### 4.4 Recommendations

From the previous discussion we now derive a set of recommendations for provable payment protocols:

- **Protocols claiming non-repudiation should specify what exactly is proved with certain pieces of evidence:**

The fact that PReq does not allow M to prove that C made a payment to M need not make the SET protocol insecure. Rather, it is important to know what can be proved and what not, and use the protocol accordingly.

- **Secure authorization using digital signatures does not guarantee provability to an external verifier unless the authorization message is sufficiently self-contained:**

C knows which are the PurchAmt and HOD in the PReq received by M; with PRes (and the CapRatio in it), C can convince itself which HOD and final amount were approved by M. In addition, C knows that M cannot prove otherwise. M, in turn, holds C's authorization of HOD and PurchAmt, and knows C cannot repudiate those.

The problem lies in the fact that PReq and PRes are linked only by TIDs, and that C cannot prove to an external verifier which PReq was received by M. This problem would not occur if PRes contained the necessary data HOD and PurchAmt, i.e., if PRes were sufficiently self-contained.

- **Hashing can be used to selectively reveal sensitive data:**<sup>3</sup>  
If PANData were hashed before using it in the calculation of HPIData, PANData would not need to be revealed to prove Auth(C-A).

## 5 *i*KP

In this section, we discuss and analyze *i*KP, with a special focus on the differences between SET and *i*KP with respect to provability.

### 5.1 *i*KP Protocol Overview

Figures 5 and 6 show the most complete version of the *i*KP protocols as implemented in the ZiP [BGH<sup>+</sup>99] prototype: 3KP with mandatory Confirm message containing the Acquirer’s signature Sig<sub>A</sub>.<sup>4</sup> For reasons of simplicity, we neglect the optional OPTSIG<sub>X</sub> parameters that parties can include in their respective signatures.

Figure 6 represents the protocol option where no signature by M is present in Invoice. (In Sect. 5.3, we will discuss the case where Sig<sub>M</sub> is present also in Invoice.)

Public-key certificates are transported outside of the *i*KP messages. The Cancel message is used if the Merchant decides not to go ahead with payment authorization and may not be used if the Merchant has sent an Auth-Request. For further details, we refer to [BGH<sup>+</sup>99].

### 5.2 Proofs of Authorizations

The rules for deriving respective authorizations from *i*KP evidence are described in [Van99]. As a general finding, *i*KP provides each of the parties with sufficient evidence to prove the other parties’ authorizations. In this section, we focus on some of the differences between *i*KP and SET.

**M proving Auth(C-M).** After receiving Payment, M can use the contents of Payment together with Common to prove Auth(C-M). M can do so without revealing DESC, the goods description, to V. This is an illustration of an *i*KP design goal, namely to disclose a minimal amount of data to achieve a certain proof. However, as DESC is present in hashed form (HDESC = H(DESC, SALT<sub>D</sub>)) in Common, M can prove DESC if necessary by revealing DESC and SALT<sub>D</sub>.

In SET, the goods description OD and amount PurchAmt were represented in the same hash HOD in PReq. Therefore, it was not possible to prove PurchAmt without revealing OD.

<sup>3</sup> This recommendation was made in [BGH<sup>+</sup>95] and its application illustrated in the discussion of *i*KP protocols in Sect. 5

<sup>4</sup> [BGH<sup>+</sup>99] uses the terms Buyer and Seller. The *i*KP notation has been changed here to be consistent with our (Customer, Merchant) notation.

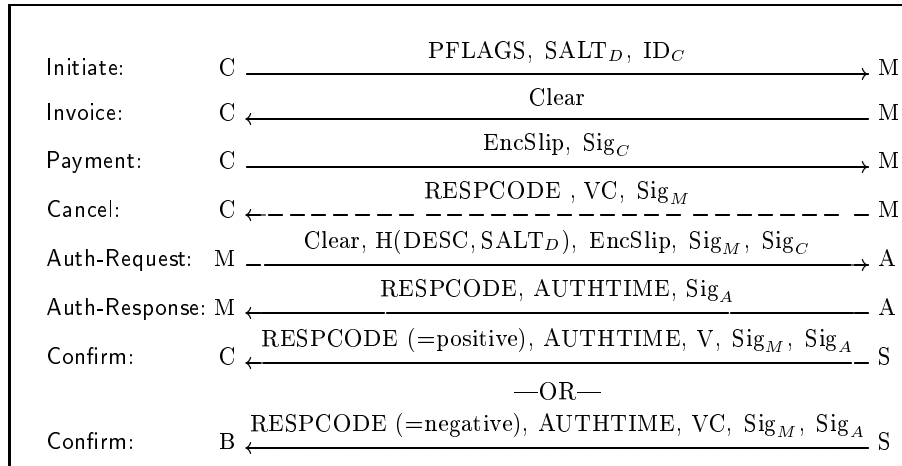
SALT <sub>D</sub>	Random number generated by C, used to salt DESC.
SALT <sub>C</sub>	Random number used to salt the account number in C's certificate.
PRICE	Amount and currency
INV-EXP	Invoice (offer) expiration specified by M.
DATE	M's date/time stamp, used for "coarse-grained" replay protection.
NONCE <sub>M</sub>	M's nonce (random number) used for more "fine-grained" replay protection.
ID <sub>M</sub>	Merchant id. This identifies M to A.
TID <sub>Z</sub>	Transaction ID assigned by each party to an <i>i</i> KP transaction. Generated at a layer above <i>i</i> KP and not explicitly carried in <i>i</i> KP messages.
DESC	Description of purchase/goods.
CAN	C's Account Number (e.g., credit card no.). Includes expiration date.
ID <sub>C</sub>	C's randomized pseudo-ID. $ID_C = H(R_C, CAN)$ with $R_C$ a random number chosen by C.
RESPCODE	Authorization response from A. Can also be set by M in the case of a cancellation.
AUTHTIME	Authorization time set by the A.
V	Random number generated by M. The combined (Sig <sub>M</sub> , V) are a proof to C that M has accepted payment.
VC	Random number generated by M. The combined (Sig <sub>M</sub> , VC) are a proof to C that M has not accepted payment.
Common	TID <sub>M</sub> , TID <sub>C</sub> , PFLAGS, PRICE, ID <sub>M</sub> , DATE, INV-EXP, NONCE <sub>M</sub> , ID <sub>C</sub> , H(DESC, SALT <sub>D</sub> ), H(V), H(VC)
Clear	PFLAGS, ID <sub>M</sub> , DATE, NONCE <sub>M</sub> , H(Common), INV-EXP, H(V), H(VC)
EncSlip	$E_A(PRICE, H(Common), CAN, R_C, SALT_C)$
Sig <sub>M</sub>	$S_M(H(Common))$
Sig <sub>C</sub>	$S_C(EncSlip, H(Common))$
Sig <sub>A</sub>	$S_A(RESPCODE, AUTHTIME, H(Common))$

**Fig. 5.** *i*KP atomic and composite fields.

**C proving Auth(M-C).** After receiving a positive Confirm (V), C can prove M's *payment* authorization of PRICE. Similarly, with a negative Confirm (VC), C can prove authorization denial (authorization with amount = 0).

In both cases, DESC associated with the authorized *payment* need not but can be revealed and proved in a similar way as in the case of M proving Auth(C-M).

The *i*KP Confirm message can thus be used as a valid and secure *payment* receipt. The *i*KP Confirm message is thus self-contained: its combined (Sig<sub>M</sub>, V) signature includes all the necessary data (H(Common)) to be proved, as opposed to the SET PRes message (which does not contain HOIData).



**Fig. 6.** 3KP payment with in-line authorization.

**A proving Auth(C-A).** A can prove Auth(C-A) by providing AuthReq and Common to V.

An interesting difference between *i*KP and SET should be mentioned here. In *i*KP, EncSlip allows A to retrieve necessary values (such as PRICE) to rebuild Common, but its contents (credit card number . . .) need not be revealed to V in order to prove Auth(C-A). In SET, A needs to reveal the decrypted contents of PI to V in order to show Auth(C-A). The reason is that A does not possess OIData and therefore needs to rely on proving the relationship between (decrypted) PIData and Sig<sub>C</sub>.

**C proving Auth(A-C).** With a Confirm with positive (negative) RESPCODE, C can prove A's positive (amount = PRICE) or negative (amount = 0) authorization of *value subtraction*. In SET, PRes does not contain a signature by A and thus the evidence collected by C does not allow C to prove Auth(A-C).

**A proving Auth(M-A).** With Auth-Request and the reconstructed Common (using values from the decrypted EncSlip), A can prove M's authorization of *value claim*.

As mentioned in Sect. 5.1, in a variant of the 3KP protocol, Sig<sub>M</sub> is sent already in Invoice, representing a signed offer. The use of this protocol option deserves a cautionary note: Sig<sub>M</sub>, included in Invoice, can be used by the Customer (or an intruder intercepting Payment) to compose a valid Auth-Request without awareness or consent by the Merchant. This may lead to scenarios where an honest Merchant M cancelled a payment but a dishonest party may make it seem that M authorized it at the same time. (Remember that M is not allowed

to send both Cancel and Auth-Request.) In [Van99], a couple of suggestions are made to solve this problem.

**M proving Auth(A-M).** With a positive Auth-Response (RESPCODE positive), M can prove A's authorization of *value claim*; a negative RESPCODE proves A's denying *value claim* authorization.

### 5.3 Discussion

In 3KP, each of the parties obtains the evidence needed for proving the other parties' authorizations. Proving authorizations also reveals less sensitive data than in SET: DESC, the description of the goods, need not be revealed unless it explicitly needs to be proved; similarly, the encrypted credit-card slip need not be revealed to the verifier to prove B's authorization of *value subtraction*.

A comment can be made about the value of signing a hash or hash tree. As shown above, all three parties sign  $H(\text{Common})$ , and their signatures commit them to all the values in  $H(\text{Common})$ . All three parties thus sign  $H(\text{DESC}, \text{SALT}_D)$ . However, only C and M know DESC. Thus only C and M commit themselves to DESC when signing  $H(\text{DESC}, \text{SALT}_D)$ : for A, signing  $H(\text{DESC}, \text{SALT}_D)$  does not entail signing its constituent parts. Such a distinction should be pointed out in the definition of the protocol and its provability rules.

## 6 Summary

In this paper, we formulated a number of provability requirements for payment protocols. Our initial assumption was that every honest participant should collect enough evidence to prove its transaction views without relying on evidence provided by other parties. This is needed to use digital receipts as self-contained pieces of evidence in off-line verification scenarios. Furthermore, relying on other parties to give evidence requires trust in those parties. Also, it assumes that these other parties' payment systems are enhanced with interoperable dispute handling and evidence collection mechanisms. This is realistic only if those mechanisms are standardized to the same level as the payment system itself, and if they become a mandatory part of it.

We validated the most secure version (where all parties digitally sign messages) of SET against these provability requirements, and concluded that some important assertions cannot be derived from evidence collected during a correct protocol run. Specifically, the Customer in a SET transaction has no secure receipt of payment. A comparison shows the equivalent version of *iKP* to provide more complete evidence than SET.

Some of the findings led to general recommendations for constructing provable payment protocols. One example is the use of hash trees, constructed such that sensitive data can be revealed selectively.

Another, more important, general finding is the following: secure authorization using digital signatures does not guarantee provability of the authorization to an external verifier unless the message is sufficiently self-contained.

In conclusion, we recommend that protocols claiming non-repudiation explicitly specify the rules to be used for deriving authorization statements from digitally signed messages.

## References

- [AHS98] N. Asokan, Els Van Herreweghen, and Michael Steiner. Towards a framework for handling disputes in payment systems. In *Third USENIX Workshop on Electronic Commerce*, pages 187–202, Boston, Mass., September 1998. USENIX. Available from <http://www.zurich.ibm.com/Technology/Security/publications/1998/AvHS98b%.ps.gz>.
- [APASW98] J. L. Abad-Peiro, N. Asokan, Michael Steiner, and Michael Waidner. Designing a generic payment service. *IBM Systems Journal*, 37(1):72–88, January 1998.
- [BGH<sup>+</sup>95] Mihir Bellare, Juan Garay, Ralf Hauser, Amir Herzberg, Hugo Krawczyk, Michael Steiner, Gene Tsudik, and Michael Waidner. iKP – A family of secure electronic payment protocols. In *First USENIX Workshop on Electronic Commerce*, pages 89–106, New York, July 1995. USENIX.
- [BGH<sup>+</sup>99] Mihir Bellare, Juan Garay, Ralf Hauser, Amir Herzberg, Hugo Krawczyk, Michael Steiner, Gene Tsudik, Els Van Herreweghen, and Michael Waidner. Design, implementation and deployment of a secure account-based electronic payment system. Research Report RZ 3137, IBM Research Division, June 1999. A modified version is to appear as [BGH<sup>+</sup>00].
- [BGH<sup>+</sup>00] Mihir Bellare, Juan Garay, Ralf Hauser, Amir Herzberg, Hugo Krawczyk, Michael Steiner, Gene Tsudik, Els Van Herreweghen, and Michael Waidner. Design, implementation and deployment of the iKP secure electronic payment system. *IEEE Journal on Selected Areas in Communications*, 18, 2000, in press.
- [Bol97] Dominique Bolognani. Towards the formal verification of electronic commerce protocols. In *10th IEEE Computer Security Foundations Workshop*, pages 133–146. IEEE Computer Press, 1997.
- [Bra97] S. Brackin. Automatic formal analyses of two large commercial protocols. In *DIMACS Workshop on Design and Formal Verification of Security Protocols*, Rutgers New Jersey, September 1997.
- [Kai95] Rajashekar Kailar. Reasoning about accountability in protocols for electronic commerce. In *Proceedings of the IEEE Symposium on Research in Security and Privacy*, Oakland, CA, May 1995. IEEE Computer Society Press.
- [Kat96] M. Ethan Katsch. Dispute resolution in cyberspace. In *Connecticut Law Review Symposium: Legal Regulation of the Internet*, number 953 in 28, 1996. Available from <http://www.umass.edu/legal/articles/uconn.html>.
- [KN98] Volker Kessler and Heike Neumann. A sound logic for analysing electronic commerce protocols. In J.-J. Quisquater, Y. Deswarte, C. Meadows, and D. Gollmann, editors, *Proceedings of the Fifth European Symposium on Research in Computer Security (ESORICS)*, number 1485 in Lecture Notes in

Computer Science, Louvain-la-Neuve, Belgium, September 1998. Springer-Verlag, Berlin Germany.

- [MS98] Catherine Meadows and Paul Syverson. A formal specification of requirements for payment transactions in the SET protocol. In *Proceedings of the Financial Cryptography Conference (FC98)*, 1998.
- [MV97] Mastercard and Visa. *SET Secure Electronic Transactions Protocol*, version 1.0 edition, May 1997. Book One: Business Specifications, Book Two: Technical Specification, Book Three: Formal Protocol Definition. Available from [http://www.setco.org/set\\_specifications.html](http://www.setco.org/set_specifications.html).
- [Van99] Els Van Herreweghen. Using digital signatures as evidence of authorizations in electronic credit-card payments. Research Report 3156, IBM Research, June 1999. available from <http://www.zurich.ibm.com/Technology/Security/publications/1999/VanHer99.ps.gz>.