iKP -- A Family of Secure Electronic Payment Protocols

Mihir Bellare and Juan A. Garay
IBM T.J. Watson Research Center

Amir Herzberg, Hugo Krawczyk, Michael Steiner,
Gene Tsudik, and Michael Waidner
IBM Zurich Research Laboratory

iKP – A Family of Secure Electronic Payment Protocols

EXTENDED ABSTRACT*

Mihir Bellare†, Juan A. Garay†, Ralf Hauser‡, Amir Herzberg†,
Hugo Krawczyk†, Michael Steiner‡, Gene Tsudik‡, Michael Waidner‡

August 2, 1995

Abstract

This paper proposes a family of protocols – iKP ($i = 1, 2, 3$) – for secure electronic payments over the Internet. The protocols implement credit card-based transactions between the customer and the merchant while using the existing financial network for clearing and authorization. The protocols can be extended to apply to other payment models, such as debit cards and electronic checks. They are based on public-key cryptography and can be implemented in either software or hardware. Individual protocols differ in key management complexity and degree of security. It is intended that their deployment be gradual and incremental.

The iKP protocols are presented herein with the intention to serve as a starting point for eventual standards on secure electronic payment.

1 Introduction

Nowadays it is hardly necessary to stress the importance of electronic commerce. Let us just note that it is rapidly gaining momentum, and is equally appealing to both (electronic) merchants and consumers.

There is widespread agreement that to enable electronic commerce one needs means for *secure electronic payments.* Indeed, the appeal of electronic commerce without electronic payment is limited. Moreover, *insecure* electronic payment methods are more likely to impede, than to promote, electronic commerce.

In this paper we propose a family of secure electronic payment protocols – iKP (i-Key-Protocol, $i = 1, 2, 3, \ldots$). The protocols are compatible with the existing business models and payment system infrastructure. They involve three parties: the customer (who will make the payment), the merchant (who will receive the payment) and a party called a *gateway* (who acts as a gateway between and electronic world and the existing payment infrastructure, and will authorize the transaction by using the existing infrastructure).

Within this framework we focus on the credit card payment model as it is anticipated to be the most popular in the near future. (The customer has a credit card. The gateway is now called an acquirer gateway and will use the existing credit card transaction verification infrastructure to authorize requests). However, the protocols can be extended to apply to other payment models, e.g., debit cards and electronic checks.

All iKP protocols are based on public-key cryptography, but they vary in the number of parties (out of the three involved parties) that possess their own public key pairs. This number is indicated by the name of the individual protocols: 1KP, 2KP, and 3KP. The iKP protocols offer increasing levels of security and sophistication as the number of parties who can hold public key pairs increases.

The simplest protocol, 1KP, requires that only the acquirer gateway possess a pair of public and private keys. Customers and merchants need only to

† IBM T.J. Watson Research Center, P.O. Box 704, Yorktown Heights, NY 10598, USA, {mihir, garay, amir, hugo}@watson.ibm.com

‡ IBM Zürich Research Laboratory, Sämerstrasse 4, CH-8803 Rüschlikon, Switzerland, {rah, sti, gts, wmi}@zurich.ibm.com

* The most recent version of this document is available from <http://www.zurich.ibm.com/Technology/Security/extern/ecommerce/>.

possess the authentic public key of the gateway, or the authentic public key of an "authority" that validates the gateway's public key via a signed certificate. This involves a minimal certification authority infrastructure that provides certificates for a small number of entities, namely, the acquirer gateways. Such a minimal certification authority can be run, for example, by the credit card company itself. (In which case, a customer will keep, say, the "VISA public key" or "MasterCard public key", etc.) In the 1KP scenario, customers are authenticated on the basis of their credit card numbers, and possibly associated secret PINs. Payments are authenticated by communicating the credit card number and PIN appropriately *encrypted* under the acquirer's public key, and properly bound to relevant information (purchase amount, id's, etc.). While 1KP is very simple, it does not offer non-repudiation for messages sent by customers and merchants; this means that disputes about the authenticity of payment orders are not easily resolvable. Some consequences of missing non-repudiation for payment systems are illustrated in [1].

2KP asks that merchants, in addition to acquirer gateways, hold public key-pairs and public key certificates. The protocol is can then provide non-repudiation for messages originated by merchants. Additionally, 2KP enables customers to verify that they are dealing with *bona fide* merchants by checking their certificates, without any on-line contact with a third party. As in 1KP, payment orders are authenticated via the customer's credit card number and PIN (encrypted before transmission).

3KP further assumes that the customer has a public key pair, and then provides full multi-party security. It achieves non-repudiation for all messages of all parties involved. Payment orders are authenticated both by the credit card number and PIN, and a digital signature of the customer. This makes the forging of payment orders computationally infeasible. Additionally, 3KP enables merchants to authenticate customers on-line. Notice that in this case a full certification authority infrastructure is required to provide certificates of the customer's public keys.

All *i*KP protocols can be implemented either in software or hardware. In fact, in 1KP and 2KP the customer does not even need a personalized payment device: only credit card data (and the PIN if present) must be entered to complete a payment. However, for the sake of increased security, it is desirable to use a tamper-resistant device that can protect the PIN and – in case of 3KP – the secret key of the customer.

As public key technology becomes more pervasive, we expect more and more parties to hold public key pairs. A gradual deployment of the *i*KP protocols thus makes sense: begin with 1KP, then move to 2KP and finally to 3KP.

We stress that the job of *i*KP is to enable *payments*. It is not concerned with any aspect of the determination of the order; it assumes that the order, including price, have already been decided on between customer and merchant.

The iKP protocols do not explicitly provide encryption of the order information. Such protection is assumed to be provided by other existing mechanisms, e.g., SHTTP [18] or SSL [14]. The decoupling of order encryption from the electronic payment protocol is an important design principle of *i*KP which supports compatibility with different underlying browsing and privacy-protecting mechanisms. It also adds to the simplicity, modularity, and ease of analysis of the protocols. An additional advantage is freeing *i*KP from export restrictions related to the use of bulk encryption. Nonetheless, if desired, the *i*KP family (especially, 2KP and 3KP) can be easily extended to generate shared keys between customer and merchant for protection of browsing and order information.

2 Related Work

The *i*KP family has many features and motivations in common with other proposals for on-line payment systems[1].

Most proposals for on-line payment are based on standard models (e.g, credit cards) and connect the electronic and the conventional payment system via some sort of gateway (e.g., [20, 9, 12]).

As already mentioned, most current on-line payments are not protected at all. Some systems propose symmetric cryptography for efficiency reasons (e.g., [11, 19, 17]), in particular, those that aim at micro-payments. However, most proposals use public key cryptography in a way similar to one or another *i*KP protocol. For example, [12] uses public-key cryptography between merchant and gateway, and the protocol sketched in [9] appears cryptographically similar to 2KP.

The most cryptographically advanced electronic payment systems emphasize *untraceability* and *anonymity* against the payment system [3, 5, 6, 4, 10, 16].

Finally, there are some general security schemes

[1] For a comprehensive listing see [15] or <http://www.zurich.ibm.com/ Technology/Security/sirene/outsideworld/ecommerce.html>.

for the *World Wide Web*, most notably, SHTTP [18] and SSL [14]. Both have been suggested as a basis for secure electronic payments. SHTTP is a possible platform for implementing *i*KP. SSL is more thought to secure the link between WWW client and server, and is therefore less suited for multi-party protocols like *i*KP. Moreover, SSL does not support non-repudiation.

The particular advantages of *i*KP over existing proposals are:

- The *i*KP family allows for a gradual deployment. 1KP is based on what already exists today – credit cards, PINs, and the existing payment system networks – and presents a feasible short-term solution. Introduction of public key cerfitication of merchants will usher in 2KP and, as soon as the certification infrastructure for customers is in place, 3KP can be phased in and achieve full multi-party payment security.
- *i*KP is an evolving design, rather than a fixed, closed protocol. It is intended as a starting point for a standard for secure payments over the Internet. It is open for discussion, and we encourage comments on its qualities and suggestions for improvement.
- The use of encryption in *i*KP is limited to well-defined payment data – credit card numbers and PINs – and the interfaces to cryptographic primitives can be designed in a way that makes them inaccessible to the end-user. Therefore, we expect that *i*KP (at least specific implementations as sketched in [13]) will not be subject to US export regulations. We note that, for countries outside North America, there is absolutely no incentive to use payment systems with restricted or weakened security.

A specific software-only architecture used for implementing a prototype of *i*KP is described in [13]. It is independent of any HTTP-extension and works with any WWW browser.

3 Payment Model

PARTIES. All *i*KP protocols are based on the existing credit-card payment system. The parties in the payment system are shown in Figure 1.

The payment system is operated by a *payment system provider* like Europay, MasterCard, VISA. This payment system provider has fixed business relations with certain banks who act as *issuers* of credit cards to customers, or as *acquirers* of payment records from merchants. Each issuer has a Bank Identification Number, BIN, which it receives at the time it signs up with a payment system provider, and which is embossed on each credit card issued as part of the credit card number. The BIN also identifies the payment system provider.

A *customer* receives a credit card from an issuer, and is in possession of a PIN as is common in current systems. In 1KP and 2KP, payments will be authenticated only by means of the credit card number and this PIN (both suitably encrypted!), while in 3KP, a digital signature is additionally used.

It is assumed that (as can be expected for electronic payment) that the customer is using a computer to execute the payment protocol. Since this computer must receive the customer's PIN or secret signature key, it must be a trustworthy device. We caution that even a customer-owned computer is vulnerable: it may be used by several persons or it may contain a Trojan horse or a virus that could steal PINs and secret keys. The best payment device would be a secure isolated computer, e.g., a tamper-resistant smartcard, connected to the computer used for shopping via a customer-owned smartcard reader with its own keyboard and display. (This is often called an *Electronic Wallet.*) Technically, 1KP and 2KP can be used with any kind of payment device, while for 3KP the customers need personal devices that store their secret signature keys and certificates.

A *merchant* signs up with the payment system provider and with a specific bank, called an *acquirer,* to accept deposits. Like a customer, a merchant needs a secure device that stores the merchant's secret keys and performs the payment protocol.

Clearing between acquirers and issuers is done using the existing financial networks.

The *i*KP protocols deal with the *payment* transaction only (i.e., the solid lines in Figure 1), and therefore involve only three parties, called C – Customer, M – Merchant, and A – Acquirer Gateway. Note that A is no acquirer in the financial sense, but a *gateway* to the existing credit card clearing/authorization network. In other words, the function of A is to serve as a front-end to the *current infrastructure that remains unchanged.*

The protocols presented here describe the core of a payment system only. Besides this, additional mechanisms are needed, e.g., for *cancelation of payment orders* and for *providing statements of account.*

PUBLIC KEYS AND CERTIFICATION. Since all *i*KP protocols are based on public-key cryptography, we need a mechanism to authenticate these public keys. We assume a *certification authority, CA*, which has a secret key, SK_{CA}. Its public counterpart, PK_{CA}, is

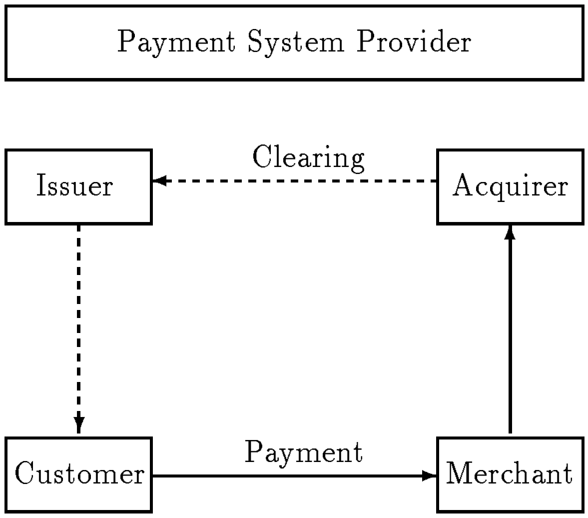

Figure 1: Generic Model of a Payment System

held by all other parties. CA will certify a public key of party X by signing the pair (X, PK_X) consisting of the identity of X and X's public key. (The signature is computed under SK_{CA}.) Note that PK_{CA} must be conveyed in an authenticated manner to every party. This will be typically done out-of-band, via any of a number of well-known mechanisms.

For simplicity's sake, we assume that there is only one certification authority. However it is easy to extend the protocols to support multiple certification authorities, e.g., such that the payment system provider at the top-level authority issues certificates to its constituent issuers and acquirers, while these, in turn, issue certificates to their customers and merchants.

In all iKP protocols, an acquirer A has a secret key, SK_A, which enables signing and decryption. Its public counterpart, PK_A, (which enables signature verification and encryption) is held by each accredited merchant together with its corresponding CA's certificate. As in current operation, acquirers receive the customers' credit card numbers and PINs, and are trusted to keep these values confidential.

In 2KP, each merchant, and in 3KP also each customer, have a secret/public key-pair. They are denoted by $(\mathrm{SK}_M, \mathrm{PK}_M)$ and $(\mathrm{SK}_C, \mathrm{PK}_C)$, respectively. Both public keys are included in certificates issued by CA.

Adversaries and Threats. We consider three different adversaries:

- **Eavesdropper** who listens to messages and tries to learn secrets (e.g., credit card numbers, PIN's)
- **Active attacker** who introduces forged messages in an attempt to cause the system to misbehave (e.g., to send him goods instead of to the customer)
- **Insider** who either is some legitimate party or learns that party's secrets. (One example is a dishonest merchant who tries to get paid without authorization.)

Before listing requirements in Section 4 we briefly discuss common threats and attacks.

The *Internet* is a decentralized, heterogeneous network, without single ownership of the network resources and functions. In particular, one cannot exclude the possibility that messages between the legitimate parties would pass through a maliciously controlled computer. Furthermore, the routing mechanisms in the Internet are not designed to protect against malicious attacks. Therefore, it is folly to assume either confidentiality or authentication for messages sent over the Internet, unless proper cryptographic mechanisms are employed. To summarize, it is easy to steal information off the Internet. Therefore, at least credit card numbers and PINs must *not* be sent in the clear.

In addition, one must be concerned about the trustworthiness of the merchants providing Internet service. The kind of business that is expected in the Internet would include the so-called *cottage industry* – small merchants. It is very easy for an adversary to set up shop and put up a fake electronic *storefront* in order to get customers' credit card numbers. This implies that the credit card number should travel from customer to acquirer without being revealed to the merchant (who needs only the BIN which can be provided separately.)

Obviously, a good deal of care must be taken to protect the keys of acquirers. One of the biggest concerns is that of an adversary breaking into an acquirer computer through the Internet connection. Therefore, the acquirer's computer must be protected with the utmost care; including a very limited Internet connection using advanced firewall technology (e.g., [8, 7].)

Furthermore, the trust in the acquirer's computer must be limited, so that a break in would have a limited effect only.

4 Security Requirements

In this section we consider a range of potential requirements for each party involved in the payment process: issuer/acquirer, merchant, and customer. They range from mandatory security requirements to optional features.

ISSUER/ACQUIRER REQUIREMENTS. The issuer and the acquirer are assumed to enjoy some degree of mutual trust. Moreover, an infrastructure enabling secure communication between these parties is already in place. Therefore, we join the requirements of the issuer and the acquirer.

A1– *Proof of Transaction Authorization by Customer.* When the acquirer debits a certain credit card account by a certain amount, the acquirer must be in possession of an unforgeable *proof* that the owner of the credit card has authorized this payment. This proof must not be "replayable," or usable as proof for some other transaction. This means it must certify at least the amount, currency, goods description, merchant identification, and delivery address, and be obtained in such a way that replay is not possible. (We use a combination of time stamps and nonces for this purpose). Note also that in this context the merchant may be an adversary, and even such a merchant must not be able to generate a fake debit. We distinguish between:

(a) *Weak Proof*, which authenticates the customer to the acquirer but does not serve as a proof for third parties, and

(b) *Undeniable Proof*, which provides full non-repudiation, i.e., can be used to resolve disputes between the customer and the payment system provider.

The same distinction will be made for all subsequently required proofs of transaction.

A2– *Proof of Transaction Authorization by Merchant.* When the acquirer authorizes a payment to a certain merchant, the acquirer must be in possession of an unforgeable *proof* that this merchant has asked that this payment be made to him.

MERCHANT REQUIREMENTS. We ask for two guarantess for the merchant.

M1– *Proof of Transaction Authorization by Acquirer.* The merchant needs an unforgeable proof that the acquirer has authorized the payment. This includes certification and authentication of the acquirer, so that the merchant knows he is dealing with the real acquirer, and certification of the actual authorization information. Note that again the amount and currency, the time and date, and information to identify the transaction must be certified. We also distinguish between (a) *Weak proof* and (b) *undeniable proof*, which provides full non-repudiation.

M2– *Proof of Transaction Authorization by Customer.* Even before the merchant receives the transaction authorization from the acquirer, the merchant might need an unforgeable proof that the customer has authenticated it. Again we distinguish between (a) *Weak Proof* and (b) *Undeniable Proof.* This requirement is necessary to provide for *off-line authorization.*

CUSTOMER REQUIREMENTS. We ask for the following guarantess to the customer who is making the payment.

C1– *Unauthorized Payment is Impossible.* It must not be possible to charge something to a customer's credit card without possession of the credit card number, PIN, and in case of 3KP, the customer's secret signature key. Thus, neither Internet rogues nor malicious merchants must be able to generate spurious transactions which end up approved by the acquirer. This must remain the case even if the customer has engaged in many prior legitimate transactions. In other words, information sent in one (legitimate) transaction must not enable a later spurious transaction. So in particular the PIN must not be sent in the clear, and not even be subject to guessing attacks! Similar to the two type of proofs of transactions, we distinguish between:

(a) *Impossibility*, which means that unauthorized payments are impossible provided the acquirer's secret key is not available to the adversary, and

(b) *Disputability*, which means that even if the acquirer's secret key is available to the adversary (e.g., because the adversary co-operates with an insider), the customer can prove that he/she did not authorize the payment.

In fact, these two requirements are typically met by meeting the corresponding acquirer requirements A1.a and A1.b, respectively.

C2– *Proof of Transaction Authorization by Acquirer.* The customer would like to be in possession of proof that the acquirer authorized the transaction. This "receipt" from the acquirer is not of paramount importance, but is convenient to have. Again, we distinguish between (a) *Weak Proof* and (b) *Undeniable Proof* (full non-repudiation).

C3– *Certification and Authentication of Merchant.* The customer needs a proof that the merchant is accredited at an acquirer (which could be considered as some guarantee for the trustworthiness of the merchant).

C4– *Receipt from Merchant.* The customer wants a proof that the merchant who has made the offer has received payment and promised to deliver the goods. This takes the form of an undeniable receipt. 2KP and 3KP will satisfy this requirement, but will not ensure fairness: The merchant can always refuse sending this receipt while already having received the authorization message from the acquirer gateway. In this case, the customer must take the next statement of account as a replacement for this receipt.

Additional possible requirements. Requirements C1 – C4 are discussed in the following. Other requirement that may be desirable, but are not explcitly addressed here; however we now discuss these requirements and their relation to iKP.

C5– *Privacy.* Customers want privacy of their order and payment information. For example a businessman may be purchasing the latest information on certain stocks and may not want competitors to know which stocks he is interested in. The privacy of order information and amount of payment should be implemented independently of the payment protocol, e.g., based on SHTTP [18] or SSL [14]. iKP does provide some privacy: it does not reveal order information to any other party than the merchant, at least as long as there is no dispute. But does not include encryption of these data. Obviously, credit card number and PIN must be protected carefully, which is achieved within iKP by encrypting them with the acquirer's public key. (This is the only application of *encryption* in iKP, which is made in order to facilitate exportability from the US.)

C6– *Anonymity.* Besides confidentiality of order and payment information, customers may want *anonymity* from eavesdroppers and (optionally) also from the merchant. It is also conceivable that the customer may even want anonymity with respect to the payment system provider. iKP supports anonymity from the merchants in the sense that the customer's identity, address, etc., is not revealed to the merchant: the customer uses a pseudo-identity CID which is different in each transaction. iKP does not offer anonymity from the payment system provider. This might be desirable for systems that aim to imitate cash, but is not essential for protocols, like iKP, that follow the credit card-based payment model.

5 The iKP Family

Primitives and keys. Figure 2 summarizes the notation for the keys held by the various parties, and the cryptographic primitives we will be using. While A's key pair must enable signature and encryption, all other key pairs need to enable signatures only. Note that signing and encryption are *independent* operations; in particular, $\mathcal{E}_X(\mathcal{S}_X(\alpha)) \neq \alpha$.

We want an encryption function $\mathcal{E}_X$ which provides some form of "message integrity." Decryption of a ciphertext results either in a plaintext message, or in a flag indicating non-validity. Formally, the primitive we has the property that correct decryption convinces the decryptor that the transmitter "knows" the plaintext that was encrypted. In particular, tampering with ciphertext is detectable. A

- **Keys:**

$\mathrm{PK}_X, \mathrm{SK}_X$	Public and secret key of Party X (X = Certification Authority CA, Customer C, Merchant M, Acquirer Gateway A).
CERT_X	Public key certificate of Party X, issued by CA. We assume it includes X, PK_X and CA's signature on PK_X.

All protocols assume A has a public key, and any party needing it has PK_{CA}. 1KP assumes no other keys; 2KP additionally assumes M has a public key; 3KP further assumes C also has a public key.

- **Cryptographic primitives:**

$\mathcal{H}(\cdot)$	A strong collision-resistant one-way hash function. Think of $\mathcal{H}(\cdot)$ as returning "random" values.
$\mathcal{E}_X(\cdot)$	Public-key encryption using PK_X, done in a way to provide not only confidentiality but also some kind of "message integrity."
$\mathcal{S}_X(\cdot)$	Signature with respect to SK_X. Note the signature of message M does NOT include M. We assume the signature function hashes the message before signing.

Figure 2: *Keys and cryptograhic primitives used in iKP protocols*

simple scheme to achieve such *plaintext-aware encryption* using RSA is described in Appendix A, based on the scheme of [2].

We stress that such encryption does not provide authentication in the manner of a signature, i.e., it does not provide non-repudiation. But it can be made to provide an authentication like capability between parties sharing a key (such as the CAN or PIN).

We note that the encryption function is *randomized:* $\mathcal{E}_X$ invoked upon message m will use, to compute its output, some randomizer, so that each encryption is different from previous ones.

A prototype implementation done at IBM Research uses RSA with key length 1024 for signature, and for the basis of the plaintext aware encryption; it also uses MD5 as a hash function.

Figure 3 is a list of quantities that will occur in the protocols. Their meaning and usage will be further explained as we go along.

FRAMEWORK OF *i*KP PROTOCOLS. The protocols have a common framework. Figure 4 illustrates the flows at a very high level. Before the protocol begins, each party $X \in \{A, C, M\}$ has some starting information represented by ST-INF$_X$. The customer starts with the public key PK_{CA} of the certification authority. The merchant has the certificate CERT_A of the acquirer, and the acquirer has his own certificate CERT_A plus the corresponding secret key SK_A. They may each also have other information, which differs depending on whether we are in 1KP, 2KP or 3KP, and will be specified at the appropriate time.

It is assumed that before the protocol starts, the customer and merchant have agreed on the description and price of the items to buy. The functionality required to shop and agree on the item and price are to be provided by the browser, not by *i*KP. Thus DESC and PRICE are part of the starting information of merchant and customer.

The protocol consists of five flows. The exact content of these flows depends on the protocol: they are different in 1KP, 2KP and 3KP. At a high level, however, there is a common structure. The customer starts with an Initiate flow. The merchant responds by providing the Invoice. The customer than makes the Payment which the merchant uses to send an authorization request Auth-Request to the acquirer. The acquirer goes through the financial network to obtain the authorization and returns an authorization response Auth-Response to the merchant. The latter processes this to produce a confirmation flow Confirm for the client.

The main difference between 1KP, 2KP and 3KP is the increasing use of signatures as more of the parties involved are able to use public keys.

5.1 1KP

1KP (illustrated in Figure 5) represents the initial step in the gradual introduction of a public-key infrastructure. Although it requires the use of public-key encryption by all parties, only the acquirer gateway, A, needs to possess and distribute its own public key certificate, CERT_A. In particular, the total number of certificates to be issued by the certification authority is small as it depends only on the number of gateways.

Like all members of the iKP Family, 1KP requires that all customers and merchants have an authentic copy of PK_{CA}, the public key of the certification authority. A customer C has a customer account number CAN known to the acquirer. This could be a credit card number. It may also have a secret PIN which is also known to the payment system (but not to the merchants!). Every merchant has to know the certificate of the corresponding acquirer gateway, CERT_A.

1KP does not assume A to keep a state per customer. Instead, the customer's PIN is verified using the existing authorization infrastructure (which uses tamper-resistant technology for processing and verification of PIN's).

All parties in 1KP must perform certain public key computations. Encryption is only applied *once,* for sending credit card data and PIN from the customer to the acquirer gateway, securely. Therefore, public key encryption is required from C only, while decryption is required from A only (this is true also for 2KP and 3KP). In 1KP, only A has to sign some data, which must be verified by C and M. We now provide the flow by flow actions of the parties.

Initiate: Customer forms CID by generating random number R_C and computing $\mathrm{CID} = \mathcal{H}(R_C, \mathrm{CAN})$. Generates another random number SALT_C to be used for "salting" the hash of merchandise description (DESC) in subsequent flows. Sets Text_0 to include desired protocol options (if any) and/or DESC. Sends Initiate.

Invoice: The computation of the second flow, Invoice, takes place as follows. Merchant retrieves SALT_C and CID from Initiate. Chooses/obtains DATE– this is a time stamp, and indicates, say the hour as well. Generates nonce NONCE_M. The combination of DATE and NONCE_M will be used later by A to uniquely identify this order: the nonce disambiguates payments with a common DATE. Chooses transaction id TID_M which identifies the context. Computes $\mathcal{H}(\mathrm{DESC}, \mathrm{SALT}_C)$. Forms Common as defined above and computes $\mathcal{H}(\mathrm{Common})$. Note: seller does not need to additionally "salt" $\mathcal{H}(\mathrm{Common})$ because it contains the already-salted $\mathcal{H}(\mathrm{DESC}, \mathrm{SALT}_C)$. Composes Text_1. (If C did not already have CERT_A then it could go here. Or this could include a context pointer for the customer.) Finally sends Invoice.

Payment: Customer retrieves Clear from Invoice. He retrieves ID_M, DATE, TID_M and NONCE_M. He already has PRICE and CID, so that he can now form Common. He computes $\mathcal{H}(\mathrm{Common})$ and checks that this matches the value in Clear. He then forms the SLIP as defined in Figure 5. (It includes the price, the customer account number (credit card number), and $\mathcal{H}(\mathrm{Common})$. It also includes the salt R_C used to form the CID, and optionally the PIN if present.) The slip is now encrypted under the acquirer public key: he sets $\mathrm{EncSlip} = \mathcal{E}_A(\mathrm{SLIP})$. This, along with the optional Text_2, is the Payment flow sent to the merchant. (Note: Customer doesn't do any check on DATE, other, perhaps, than making sure it is not in the future! The DATE goes into Common, and SLIP contains $\mathcal{H}(\mathrm{Common})$, and the acquirer will check the latter.)

Auth-Request: The merchant will now ask check that the acquirer authorizes the payment. He forwards EncSlip. He also sends Clear and $\mathcal{H}(\mathrm{DESC}, \mathrm{SALT}_C)$, and optional Text_3.

Auth-Response: The acquirer gateway extracts Clear, $\mathcal{H}(\mathrm{DESC}, \mathrm{SALT}_C)$ and EncSlip from Auth-Request. It then does the following:

(1) Extracts from Clear the following– ID_M, TID_M, DATE, NONCE_M and the value h_1 which is supposed to be $\mathcal{H}(\mathrm{Common})$. It now checks for replays. That is, it makes sure that there is no previously processed request with these values of ID_M, TID_M, DATE and NONCE_M.

(2) Now it decrypts EncSlip. If the decryption fails, then the alteration of EncSlip (by an adversary or by M) is detected and the transaction is invalid. If not, A gets SLIP. Now A extracts PRICE, the value h_2 which is supposed to be $\mathcal{H}(\mathrm{Common})$, CAN, R_C, and, if present, the PIN from SLIP.

(3) It checks that $h_1 = h_2$– this ensures that merchant and customer agree on the order information (price, identity of merchant, etc).

(4) It re-forms Common. (It has PRICE from SLIP. It has ID_M, TID_M, DATE, and NONCE_M from Clear. It can compute $\mathrm{CID} = \mathcal{H}(\mathrm{CAN}, R_C)$ because it has R_C and CAN

from SLIP. Finally it has $\mathcal{H}(\mathrm{DESC}, \mathrm{SALT}_C)$ from Auth-Request. These put together yield Common.) It then computes $\mathcal{H}(\mathrm{Common})$ and checks this equals the value $h_1 = h_2$ above.

(5) Now it uses the existing clearing and authorization system to on-line authorize the payment: for this, it will forward CAN, PIN if present, the price, etc as dictated by the authorization system. Upon receipt of a response Y/N from the authorization system, A computes a signature, using the function $\mathcal{S}_A$, on Y/N and $\mathcal{H}(\mathrm{Common})$.

Finally it sends Auth-Response and possibly Text_4. The latter could include TID_M so that the merchant can easily recover the context.

Confirm: M receives Auth-Response. He extracts Y/N and the acquirer signature. He already has $\mathcal{H}(\mathrm{Common})$. Now he checks that the acquirer sent a valid signature of $\mathrm{Y/N}, \mathcal{H}(\mathrm{Common})$. He then forwards Y/N to the customer. He also forwards the acquirer signature so that the customer may check it.

There are some final checks by the buyer: for example it may want to check the acquirer signature. We stress here that the use of $\mathcal{H}(\mathrm{Common})$ in the signature (as opposed to using the explicit values amount, currency, etc.), is done in order to protect the privacy of these data when transmitted to merchant and customer. We now look at which requirements 1KP satisfies.

A1(a) *Proof of Transaction Authorization by Customer.* SLIP includes the CAN and the PIN. (The latter, if present, is known only to the customer and payment system and is the basis of the security. If it is not present, one must assume the CAN is not known to an adversary.) Since C knows PK_{CA} and verifies CERT_A, it is ensured that C does not unwittingly send the CAN and PIN to a non-authorized party. A decrypts and checks that the CAN and PIN are correct. The plaintext-awareness of the encryption (see beginning of Section 5) implies that SLIP originated with the CAN and PIN holder. An adversary not knowing the CAN or PIN can neither create a fake SLIP nor modify the encryption of a legitimate one to its advantage.

Replay of a SLIP by a dishonest merchant will be detected by the combination of the DATE and NONCE_M. There is an "acceptable delay" period T_{delay}. Slips containing a particular DATE are kept until for T_{delay} more time than that indiacated by DATE. (For example, DATE could be the date and hour, and the delay period a day, meaning slips are kept for a day more than the DATE marked on them.) Within a particular value of DATE, different slips are disambiguated by the nonces.

The "semantic security" of the encryption (and in particular the fact that it is randomized) implies also security against *dictionary-attacks.* If the attacker knows all data in SLIP except PIN, he could compute encryptions $\mathcal{E}_A(\mathrm{SLIP})$ for *all* possible values of PIN. With a deterministic encryption function, he could easily determine the correct PIN by comparing all encryptions with the one produced by C. Therefore, plain-text aware encryption is randomized: if SLIP is encrypted twice, two *different* cyphertexts are produced, which excludes this type of attack.

Note that PIN-based authentication provides a weak proof only. Signature-based authentication as used in 3KP provides an undeniable proof. Moreover, the probability of guessing the correct PIN is much higher than the probability of guessing a valid-looking signature.

It is important to stress that the "transaction" that of which we want a proof includes the item description, and in particular the delivery address. It should not be possible for an adversary to divert a legitimate payment by changing the delivery address. The inclusion of $\mathcal{H}(\mathrm{Common})$ in A's authorization is to prevent such attacks. In particular it prevents a certain kind of *man-in-the-middle* attack that we now describe.

An attacker that impersonates a merchant can get the agreement of the customer to buy something for a given amount. The adversary gets from the customer an encrypted slip authorizing the payment. The adversary now impersonates the customer to the merchant, but this time the adversary buys for the same amount a (possibly) different merchandise with different delivery address and "pays" for it with the customer's slip. Notice, however, that in this case there will be a mismatch between the view of the "order" by the real customer and the merchant, and, consequently, a mismatch in the value of $\mathcal{H}(\mathrm{Common})$.

M1(b): *Proof of Transaction Authorization by Acquirer.* The unforgeable, undeniable proof is the digitally-signed message sent by A. Notice that we have used a digital signature so that non-repudiability is provided. The inclusion of $\mathcal{H}(\mathrm{Common})$ prevents the replay of authorization messages which would result in fake authorization of customer's orders.

Since the merchant knows Common in advance, the signature would indicate any tampering in the information sent from merchant to acquirer, and any disagreement between customer and merchant on the payment data.

The inclusion of $\mathcal{H}$(Common) both in the customer-generated SLIP and directly in Auth-Request by the merchant enables A to detect a disagreement between merchant and customer with respect to the order contents (even before submitting the transaction to the clearing network).

C1(a): *Unauthorized Payment is Impossible.* This is a direct consequence of the achievement of A1(a).

C2(b): *Proof of Transaction Authorization by Acquirer.* As for **M1**(b).

C5: *Privacy.* Some partial privacy is provided. Specifically, the acquirer is not given DESC, but rather $\mathcal{H}(\mathrm{DESC}, \mathrm{SALT}_C)$. Furthermore, the acquirer, or an eavesdropper on the acquirer to merchant link, cannot obtain DESC via a dictionary attack, as we now explain.

In a dictionary attack, the attacker has some small set of possible values of DESC, and want to see whether one of them is what the customer is ordering. Had we not used the salt, but just sent $\mathcal{H}$(DESC), the attacker could easily make the check by evaluating $\mathcal{H}$ on his values and seeing whether one of the results matches the value $\mathcal{H}$(DESC) in the flow. But now he cannot do this because he doesn't know SALT_C. Of course, if he was powerful enough to obtain SALT_C of the C to M link (where it was transmitted in the clear) he would be able to do the dictionary attack, but that he can eavesdrop like that on both links is not too likely. Also, if privacy is really a concern, the C to M communication may be protected by SSL or SHTTP.

We stress that provision of privacy is not a primary concern of a payment protocol. However, we wish at least to not give anything away that should not be, and took the chance to add whatever privacy we could add without much cost.

The last flow from merchant to customer in which the signed authorization by the acquirer is transmitted is optional. It only serves as a receipt for the customer but is not needed for the security of the payment protocol.

To summarize, 1KP is a simple and efficient protocol whose main achievement is to get a secure electronic payment system with as little modification as possible to the existing infrastructure. Its main weaknesses are: 1) the customer authenticates itself via the acquirer and only using a credit card number and PIN (as opposed to a strong authentication via a digital signature); 2) the merchant does not directly authenticate itself to the customer or acquirer (there is some level of indirect authentication via the customer's SLIP and the authorization by the acquirer); and 3) neither merchant nor customer provide undeniable receipts for the transaction. Upgrading 1KP to provide these missing features results in the protocols described in the next two subsections, namely, 2KP and 3KP.

5.2 2KP

The second protocol, 2KP, is illustrated in Figure 6. The basic difference with respect to 1KP is that, in addition to A, each merchant M needs to possess a public key with a matching secret key, and distribute its own public key, with its certificate, CERT_M.

We now describe the additions to the flows and actions. There are two new elements in Invoice. The first is that the merchant chooses a random V and puts $\mathcal{H}(V)$ in Invoice. (The inclusion of V in Confirm will later serves as a "signature" thereby saving the merchant one signature computation. See below.) This value will be added to Common for what follows. Second, the merchant signs (using SK_M) the pair of strings $\mathcal{H}$(Common) and $\mathcal{H}(V)$ and includes this signature Sig_M in Invoice too. Furthermore the merchant includes CERT_M so that the customer can check his signature. Upon receipt of Invoice the customer checks the merchant signature, and then proceeds as before to generate Payment. Auth-Request is augmented by the merchant to include the same signature Sig_M he sent to the customer earlier, together with CERT_M. The acquirer checks this signature before authorizing payment. Finally, the value V is included by the merchant in Confirm. The customer computes $\mathcal{H}(V)$ and checks that it matches the value sent earlier in Invoice.

2KP satisfies all the requirements addressed by 1KP as well as:

A2: *Certification and Authentication of Merchant.* This is achieved by the inclusion of the merchant signature Sig_M and certificate CERT_M, and the acquirer's verification of these.

C3: *Certification and Authentication of Merchant.* Similarly achieved by inclusion of signature of merchant and its check by customer.

C4: *Receipt from merchant.* This is achieved by the combination of M's signed message sent to A, A's signed authorization message, and the value V sent in confirm. V assures the customer (and any third party) that the merchant has accepted the authorization response. (This is the payment if Y/N is yes, and the statement of rejection otherwise.) This is because no other party is capable of finding V. (It would require inverting the one-way function $\mathcal{H}$ on the point $\mathcal{H}(V)$.) Note it is important here that the customer check Sig_A – else an adversary can flip Y/N after the merchant sends Y/N and V. Thus the combination of Sig_M, V and Sig_A give the buyer undeniable proof of the seller's agreement to the transaction. The same could be achieved by M signing A's authorization message, but at the cost of an additional signature.

Obviously, M can refuse forwarding A's authorization message to the customer and sending its last message. In this case, C does not know whether the transaction was aborted or finalised (this must be handled based on the next statement of account).

5.3 3KP

As can be expected, in the last protocol – 3KP – all protocol participants, including customers, possess a public key, with the associated secret key and certificate. As illustrated in Figure 7, all parties are now able to provide non-repudiation.

The CERT_C sent to the merchant may not only contain the customer's public key and ID, but also further data. This further data is included in the certificate *in hashed form* in order to be able to reveal it to the merchant only on demand. For instance, CERT_C might include the hash of the Customer's physical address, and if ordered goods should be sent to C's home address, C can reveal "Customer's physical address" to the merchant who can verify it based on CERT_C.

The customer's signature serves as undeniable proof of transaction (A1.b), and enables disputability (C1.b). On the other hand, the merchants can link all payments of the customer with CERT_C and C's signature, i.e., the customer loses some of the privacy compared to 1KP and 2KP. One way to avoid this is by encrypting CERT_C and the signature with A's public key.

Notice that in 3KP the use of PIN numbers is only for compatibility with the existent infrastructure. Except for that reason, PINs can be safely omitted since the level of authentication provided by the customer signature is significantly superior to that provided by a PIN.

3KP satisfies all the requirements addressed by 2KP as well as:

A1(b): *Undeniable Proof of Transaction Authorization by Customer.* The customer signs the SLIP using a secret key SK_C known to C only.

M2(b): *Proof of Transaction Authorization by Customer.* Based on C's signature, M can verify that SLIP was signed by C. M cannot verify the correctness of the contents of SLIP, especially not of the PIN.

C1(b): *Unauthorized Payment is Impossible.* Follows from A1.b.

6 Summary and Comparison of the Protocols

The iKP protocols vary in the degree of both protection and complexity. They proceed in an incremental path towards electronic payment with strong security features with respect to all parties involved. Practically speaking, it is envisaged that 1KP will represent a short-term, interim step towards payment protocols with stronger security guarantees. Thereafter, 2KP and 3KP can be gradually phased in. Table 1 presents a comparison of the iKP protocols.

The iKP family can fulfill all stated requirements and, in particular, provide non-repudiatable receipts from the acquirer gateway to the merchant/customer, and from the merchant to the customer. In case that the customer also possesses a public-key pair (3KP), non-repudiation becomes possible also from the customer to the merchant/acquirer.

The protocols do not reveal the identity of the customer to the merchant. Order privacy against eavesdroppers could be achieved by applying a secure communication protocol (e.g., SHTTP [18] or SSL [14]), or, if desired, the iKP protocols themselves could be extended to provide that protection. Since iKP aims at credit-card-like payments, no anonymity against the payment system is provided. Adding anonymity and privacy to all payments is a major change in "payment culture" and only after the deployment of iKP-like systems will it be assessable whether the involved parties are inclined to move further into this direction.

The iKP protocols can be extended to support batch processing of payments from the same customer by the merchant, or to guarantee amounts as commonly done, for example, in the case of car rentals.

The protection of the acquirer from the Internet is another important aspect to the acceptability of such payment systems - first designs to minimize this exposure have been accomplished.

REQUIREMENTS/PROTOCOLS	1KP	2KP	3KP
Issuer/Acquirer			
A1. Proof of Transaction Authorization by Customer	$\surd$	$\surd$	$\surd\surd$
A2. Proof of Transaction Authorization by Merchant		$\surd\surd$	$\surd\surd$
Merchant			
M1. Proof of Transaction Authorization by Acquirer	$\surd\surd$	$\surd\surd$	$\surd\surd$
M2. Proof of Transaction Authorization by Customer			$\surd\surd$
Customer			
C1. Unauthorized Payment is Impossible	$\surd$	$\surd$	$\surd\surd$
C2. Proof of Transaction Authorization by Acquirer	$\surd\surd$	$\surd\surd$	$\surd\surd$
C3. Certification and Authentication of Merchant		$\surd\surd$	$\surd\surd$
C4. Receipt from Merchant		$\surd\surd$	$\surd\surd$

Table 1: Comparison of the iKP payment protocols. A requirement marked by $\surd$ is satisfied but not disputable, while $\surd\surd$ indicates that the requirement is satisfied based on an undeniable proof, providing non-repudiation and disputability.

References

[1] R. Anderson. Why Cryptosystems Fail. *Communications of the ACM 37/11 (1994) 32-41.* `<ftp://ftp.cl.cam.ac.uk/users/rja14/wcf.ps.Z>`.

[2] M. Bellare, P. Rogaway. Optimal Asymmetric Encryption. *Proceeding of Eurocrypt '94*, May 1994.

[3] J.-P. Boly *et al.* The ESPRIT Project CAFE - High Security Digital Payment Systems. *ESORICS '94*, LNCS 875, Springer-Verlag, Berlin 1994, 217-230. <http://www.informatik.uni-hildesheim.de/FB4/Projekte/sirene/projects/cafe/index.html>.

[4] H. Bürk, A. Pfitzmann. Digital Payment Systems Enabling Security and Unobservability. *Computers & Security*, 8/5 (1989), 399-416. <http://www.informatik.uni-hildesheim.de/ FB4/ Projekte/ sirene/lit/abstr89.html/#BuePf_89">.

[5] D. Chaum. Privacy Protected Payments. *SMARTCARD 2000, Proceedings, North-Holland, Amsterdam 1989, 69-93.*

[6] D. Chaum. Achieving Electronic Privacy. *Scientific American*, August 1992, 96-101.

[7] P. Cheng, J. Garay, A. Herzberg, and H. Krawczyk. Design and implementation of modular key management protocol and IP Secure Tunnel on AIX. In *Proc. 5th USENIX UNIX Security Syposium*, Salt Lake City, Utah, June 1995. Also available from ftp site software.watson.ibm.com: /pub/security/mkmp-ipst-usenix.ps.

[8] W. R. Cheswick, S. M. Bellovin. Firewalls and Internet Security: Repelling the Willy Hacker. *Addison Wesley, 1994.*

[9] CyberCash. The CyberCash(tm) System - How it Works. <http://www.cybercash.com/cybercash/cyber2.html>.

[10] DigiCash. About ecash. <http://www.digicash.com/ecash/ecash-home.html>.

[11] S. Dukach. SNPP: A Simple Network Payment Protocol. *Computer Security Applications Conference,1992.* <ftp://ana.lcs.mit.edu/pub/snpp/ snpp-paper.ps>.

[12] D. K. Gifford, L. C. Stewart, A. C. Payne, G. W. Treese. Switches for Open Networks. *IEEE COMPCON*, March 95. <http://www.openmarket.com/about/technical/>.

[13] R. Hauser, M. Steiner. Generic Function Extension of WWW Browsers. *IBM Research Division, Zurich Research Lab, March 1995.*

[14] K. E. B. Hickman. Secure Socket Library. *Netscape Communications Corp., Feb. 9th, 1995* <http://www.mcom.com/info/SSL.html>.

[15] P. Janson, M. Waidner. Electronic Payment over Open Networks. *Zeitschrift der Schweizerischen Informatikorganisationen xxxx (1995) xxxx-xxxx.* <http://www.zurich.ibm.com/Technology/Security/extern/ecommerce/>.

[16] S. H. Low, N. F. Maxemchuk, S. Paul. Anonymous Credit Cards. *2nd ACM Conference on Computer and Communication Security, Fairfax 1994.* <http://www.research.att.com/index.html#acc>.

[17] C. Neuman, G. Medvinsky. Requirements for Network Payment: The NetCheque Perspective. *IEEE COMPCON*, March 95. <ftp://prospero.isi.edu/pub/papers/security/netcheque-requirements-compcon95.ps.Z>.

[18] E. Rescorla, A. Schiffman. The Secure HyperText Transfer Protocol. Internet Draft. *Enterprise Integration Technologies, December 1994.* <http://www.eit.com/projects/s-http/>.

[19] M. Sirbu, J. D. Tygar. NetBill: An Internet Commerce System. *IEEE COMPCON, March 95.* <http://www.ini.cmu.edu/netbill/CompCon.html>.

[20] L. H. Stein, E. A. Stefferud, N. S. Borenstein, M. T. Rose. The Green Commerce Model. *First Virtual Holdings Incorporated, October, 1994.* <http://www.fv.com/tech/ green-model.html>.

A The encryption function

Here we describe the preferred way to get the function $\mathcal{E}_A$ using which the customer encrypts the SLIP under the acquirer's public key PK_A. We will use RSA. We let $f(x) = x^e \pmod N$ denote the RSA function and $f^{-1}(y) = y^d \pmod N$ its inverse, where N is a 1024 bit modulus. The issue is that simply encrypting under RSA– ie. setting $\mathcal{E}(x) = f(x)$– is not enough: this doesn't provide the "integrity" or "plaintext awareness" we need. Instead, we will first "embed" a up to 896-bit plaintext into a 1024 bit string r in a very special way and then compute $f(r)$. The scheme we now describe is a simplification of one in [2]. It makes use, in addition to RSA, of the hash function $\mathcal{H}$, and is provably secure assuming $\mathcal{H}$ behaves like a "random function."

Given a data string DATA, encode it into exactly $896 = 1024\text{-}128$ bits. (This means pad and include length of original data if necessary.) Then encrypt as follows:

(1) Let $x = 0^{64} \,.\, \mathrm{DATA}$. (So x has length 960 bits)

(2) Pick a random SALT of length 64 bits.

(3) Let $a = x \oplus H_1(\mathrm{SALT})$ and then let $b = \mathrm{SALT} \oplus H_2(a)$. Here $H_1()$ is a hash function outputting 960 bits and $H_2()$ is a hash function outputting 64 bits. Example ways to compute them starting from given $\mathcal{H}$ are given below.

(4) Let $r = a \,.\, b$ (this is 1024 bits) and output $f(r)$.

It is important to check the redundancy 0^{64} upon decrypting. You recover $r = a.b$, and then compute $\mathrm{SALT} = H_2(a) \oplus b$ and $x = H_1(\mathrm{SALT}) \oplus a$. If x doesn't have 64 leading zeros then reject.

For the hash functions H_1, H_2 one could use:

- $H_2(\mathrm{Text})$ = First 64 bits of output of $\mathcal{H}(\mathrm{Text})$.
- $H_1(\mathrm{Text})$ = First 960 bits of the sequence $\mathcal{H}(0, \mathrm{Text}).\mathcal{H}(1, \mathrm{Text}).\mathcal{H}(2, \mathrm{Text})....$

- **Quantities occuring in all three protocols:**

SALT_C	Random number generated by C; used to salt DESC and thus ensure privacy of DESC on the M to A link
PRICE	Amount and currency
DATE	Merchant's date/time stamp, used for "coarse grained" replay protection of a payment
NONCE_M	Merchant's nonce (counter or random number) used for more "fine grained" replay protection of a payment
ID_M	Merchant id. This identifies merchant to acquirer.
TID_M	Transaction ID. This is an identifier chosen by merchant which uniquely identifies the context
DESC	Description of purchase/goods, and delivery address. Includes payment information such as credit card name, bank identification number, and currency.
CAN	Customer's Account Number (Eg. credit card No.) Includes expiration date.
R_C	Random number chosen by C to form CID.
CID	A customer pseudo-ID which uniquely identifies C; computed as CID = $\mathcal{H}(R_C, \mathrm{CAN})$.
Y/N	Response from the clearing network: YES/NO or authorization code.
Text_j	For $j = 0, 1, 2, \ldots$. This is optional information that can accompany the flows. For example, can be used to carry context identifiers.

- **Quantities occuring in some of the protocols:**

PIN	Customer PIN which, if present, can optionally be used in 1KP to enhance the security
V	Random number generated by merchant in 2KP and 3KP for use as a proof that merchant has accepted payment

Figure 3: *Definitions of atomic fields used in iKP protocols. Composite fields formed from these are discussed later.*

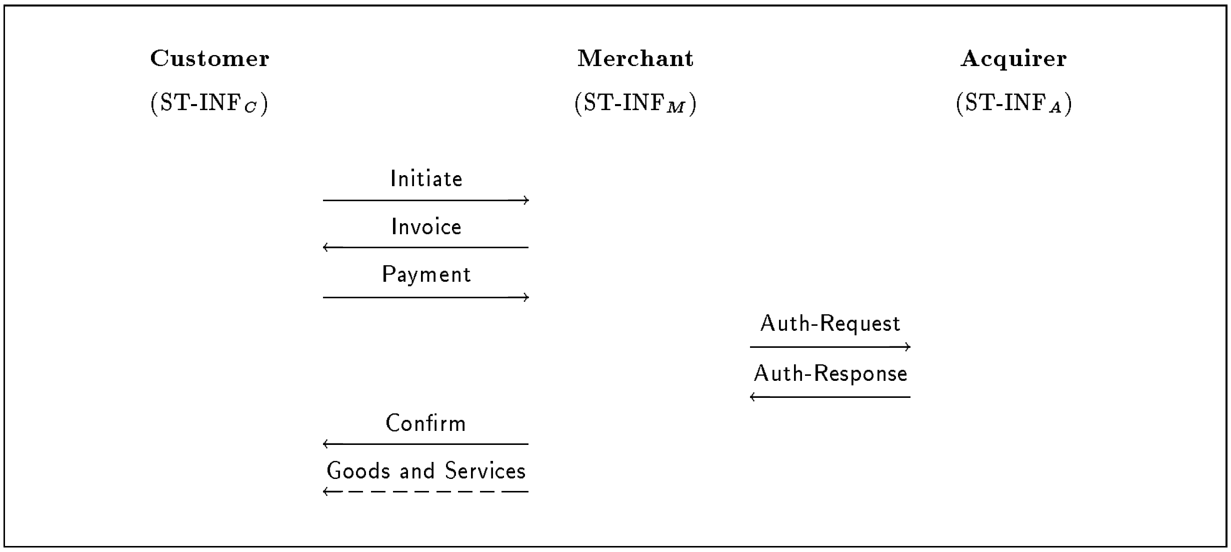

Figure 4: *Framework of iKP protocols*

- **Composite Fields:**

Common	PRICE, ID_M, TID_M, DATE, NONCE_M, CID, $\mathcal{H}(\mathrm{DESC}, \mathrm{SALT}_C)$
Clear	ID_M, TID_M, DATE, NONCE_M, $\mathcal{H}(\mathrm{Common})$
SLIP	PRICE, $\mathcal{H}(\mathrm{Common})$, CAN, R_C, [PIN].
EncSlip	$\mathcal{E}_A(\mathrm{SLIP})$

- **Starting information of parties:**

$\mathrm{ST\text{-}INF}_C$	$\mathrm{DESC}, \mathrm{CAN}, \mathrm{PK}_{CA}, [\mathrm{PIN}]$
$\mathrm{ST\text{-}INF}_M$	$\mathrm{DESC}, \mathrm{PK}_{CA}, \mathrm{CERT}_A$
$\mathrm{ST\text{-}INF}_A$	$\mathrm{SK}_A, \mathrm{CERT}_A$

- **Protocol Flows:**

Initiate: $\mathrm{C} \xrightarrow{\quad \mathrm{SALT}_C,\ \mathrm{CID},\ [\mathrm{Text}_0] \quad} \mathrm{M}$

Invoice: $\mathrm{C} \xleftarrow{\quad \overbrace{\mathrm{ID}_M,\ \mathrm{TID}_M,\ \mathrm{DATE},\ \mathrm{NONCE}_M,\ \mathcal{H}(\mathrm{Common})}^{\mathrm{Clear}},\ [\mathrm{Text}_1] \quad} \mathrm{M}$

Payment: $\mathrm{C} \xrightarrow{\quad \overbrace{\mathcal{E}_A(\mathrm{SLIP})}^{\mathrm{EncSlip}},\ [\mathrm{Text}_2] \quad} \mathrm{M}$

Auth-Request: $\mathrm{M} \xrightarrow{\quad \mathrm{Clear},\ \mathcal{H}(\mathrm{DESC}, \mathrm{SALT}_C),\ \mathrm{EncSlip},\ [\mathrm{Text}_3] \quad} \mathrm{A}$

Auth-Response: $\mathrm{M} \xleftarrow{\quad \mathrm{Y/N},\ \overbrace{\mathcal{S}_A(\mathrm{Y/N}, \mathcal{H}(\mathrm{Common}))}^{\mathrm{Sig}_A},\ [\mathrm{Text}_4] \quad} \mathrm{A}$

Confirm: $\mathrm{C} \xleftarrow{\quad \mathrm{Y/N},\ \mathrm{Sig}_A,\ [\mathrm{Text}_5] \quad} \mathrm{M}$

Figure 5: 1KP Protocol

- **Composite Fields:**

Common	PRICE, ID_M, TID_M, DATE, NONCE_M, CID, $\mathcal{H}(\mathrm{DESC}, \mathrm{SALT}_C)$, $\mathcal{H}(V)$
Clear	ID_M, TID_M, DATE, NONCE_M, $\mathcal{H}(V)$, $\mathcal{H}(\mathrm{Common})$
SLIP	PRICE, $\mathcal{H}(\mathrm{Common})$, CAN, R_C.
EncSlip	$\mathcal{E}_A(\mathrm{SLIP})$
Sig_M	$\mathcal{S}_M(\mathcal{H}(\mathrm{Common}), \mathcal{H}(V))$

- **Starting information of parties:**

$\mathrm{ST\text{-}INF}_C$	$\mathrm{DESC}, \mathrm{CAN}, \mathrm{PK}_{CA}$
$\mathrm{ST\text{-}INF}_M$	$\mathrm{DESC}, \mathrm{PK}_{CA}, \mathrm{CERT}_A, \mathrm{SK}_M, \mathrm{CERT}_M$
$\mathrm{ST\text{-}INF}_A$	$\mathrm{PK}_{CA}, \mathrm{SK}_A, \mathrm{CERT}_A$

- **Protocol Flows:**

Initiate: $\quad C \xrightarrow{\mathrm{SALT}_C,\ \mathrm{CID},\ [\mathrm{Text}_0]} M$

Invoice: $\quad C \xleftarrow{\overbrace{\mathrm{ID}_M,\ \mathrm{TID}_M,\ \mathrm{DATE},\ \mathrm{NONCE}_M,\ \mathcal{H}(V),\ \mathcal{H}(\mathrm{Common})}^{\mathrm{Clear}},\ \overbrace{\mathcal{S}_M(\mathcal{H}(\mathrm{Common}), \mathcal{H}(V))}^{\mathrm{Sig}_M},\ \mathrm{CERT}_M,\ [\mathrm{Text}_1]} M$

Payment: $\quad C \xrightarrow{\overbrace{\mathcal{E}_A(\mathrm{SLIP})}^{\mathrm{EncSlip}},\ [\mathrm{Text}_2]} M$

Auth-Request: $\quad M \xrightarrow{\mathrm{Clear},\ \mathcal{H}(\mathrm{DESC}, \mathrm{SALT}_C),\ \mathrm{EncSlip},\ \mathrm{Sig}_M,\ \mathrm{CERT}_M,\ [\mathrm{Text}_3]} A$

Auth-Response: $\quad M \xleftarrow{\mathrm{Y/N},\ \overbrace{\mathcal{S}_A(\mathrm{Y/N}, \mathcal{H}(\mathrm{Common}))}^{\mathrm{Sig}_A},\ [\mathrm{Text}_4]} A$

Confirm: $\quad C \xleftarrow{\mathrm{Y/N},\ V,\ \mathrm{Sig}_A,\ [\mathrm{Text}_5]} M$

Figure 6: 2KP Protocol

- **Composite Fields:**

Common	PRICE, ID_M, TID_M, DATE, NONCE_M, CID, $\mathcal{H}(\text{DESC}, \text{SALT}_C)$, $\mathcal{H}(V)$
Clear	ID_M, TID_M, DATE, NONCE_M, $\mathcal{H}(V)$, $\mathcal{H}(\text{Common})$
SLIP	PRICE, $\mathcal{H}(\text{Common})$, CAN, R_C.
EncSlip	$\mathcal{E}_A(\text{SLIP})$
Sig_M	$\mathcal{S}_M(\mathcal{H}(\text{Common}), \mathcal{H}(V))$
Sig_C	$\mathcal{S}_C(\text{EncSlip}, \mathcal{H}(\text{Common}))$

- **Starting information of parties:**

ST-INF_C	$\text{DESC}, \text{CAN}, \text{PK}_{CA}, \text{SK}_C, \text{CERT}_C$
ST-INF_M	$\text{DESC}, \text{PK}_{CA}, \text{CERT}_A, \text{SK}_M, \text{CERT}_M$
ST-INF_A	$\text{PK}_{CA}, \text{SK}_A, \text{CERT}_A$

- **Protocol Flows:**

Initiate: $\text{C} \xrightarrow{\text{SALT}_C,\ \text{CID},\ \text{CERT}_C,\ [\text{Text}_0]} \text{M}$

Invoice: $\text{C} \xleftarrow[\ ]{\overbrace{\text{ID}_M,\ \text{TID}_M,\ \text{DATE},\ \text{NONCE}_M,\ \mathcal{H}(V),\ \mathcal{H}(\text{Common})}^{\text{Clear}},\ \overbrace{\mathcal{S}_M(\mathcal{H}(\text{Common}), \mathcal{H}(V))}^{\text{Sig}_M},\ [\text{Text}_1]} \text{M}$

Payment: $\text{C} \xrightarrow{\overbrace{\mathcal{E}_A(\text{SLIP})}^{\text{EncSlip}},\ \overbrace{\mathcal{S}_C(\text{EncSlip}, \mathcal{H}(\text{Common}))}^{\text{Sig}_C},\ [\text{Text}_2]} \text{M}$

Auth-Request: $\text{M} \xrightarrow{\text{Clear},\ \mathcal{H}(\text{DESC}, \text{SALT}_C),\ \text{EncSlip},\ \text{Sig}_M,\ \text{Sig}_C,\ [\text{Text}_3]} \text{A}$

Auth-Response: $\text{M} \xleftarrow{\text{Y/N},\ \overbrace{\mathcal{S}_A(\text{Y/N}, \mathcal{H}(\text{Common}))}^{\text{Sig}_A},\ [\text{Text}_4]} \text{A}$

Confirm: $\text{C} \xleftarrow{\text{Y/N},\ V,\ \text{Sig}_A,\ [\text{Text}_5]} \text{M}$

Figure 7: 3KP Protocol