Journal of Universal Computer Science, vol. 13, no. 4 (2007), 479-503
submitted: 21/1/07, accepted: 27/4/07, appeared: 28/4/07 © J.UCS

Internet Payment System: A New Payment System for Internet Transactions

Zoran Đurić
(Faculty of Electrical Engineering, University of Banjaluka, Bosnia and Herzegovina
zoran@spinter.net)

Ognjen Marić
(Faculty of Mathematics and Natural Sciences, University of Banjaluka, Bosnia and Herzegovina
omar@teol.net)

Dragan Gašević
(School of Computing and Information Systems, Athabasca University, Canada
dgasevic@acm.org)

Abstract: Payment systems need to address a number of security issues in order to be an effective and secure means of transferring payments across the Internet. To be accessible to a wider audience, they also need to be easy to use for their end-users (customers and merchants).

Trying to address these issues, we created the Internet Payment System (IPS). IPS tries to combine the advantages of several existing payment systems. While strong emphasis is made on the mobility and ease of use for its customers, IPS still retains strong security properties. It achieves privacy, integrity, authentication and non-repudiation by using different cryptographic algorithms and techniques. To demonstrate that the protocol satisfies the desired security properties, we use a recently proposed tool for formal verification, called AVISPA.

Keywords: payment systems, security, cryptography, e-commerce, formal verification
Categories: C.2.2, C.2.4, D.4.6, K.4.4, K.6.5

1 Introduction

The Internet has become an essential tool for commerce and financial services. With the help of new communication technologies, these services have experienced tremendous growth. They are becoming more and more accessible to customers, regardless of their location. An inhibiting factor for this growth is the fear of fraud and sensitive data theft, which is widespread among the general public due to the insecure and unreliable nature of the Internet.

In this paper, we focus on the following e-commerce scenario: a customer wishes to purchase goods online; the payment is made by the means of a credit card, and the goods need to be shipped physically. This scenario assumes the existence of three participants: a customer, a merchant, and a financial institution (e.g. a bank). We refer to these three participants as C, M, and B, respectively. These participants are connected with communication links as shown in Figure 1. In order to perform the purchase, the participants need to exchange certain data over those links.

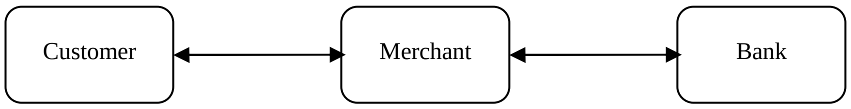

Figure 1: A typical e-commerce scenario

If the data is transmitted over the links in plain text, there is a possibility of eavesdropping. Anyone listening to the network traffic could gain access to sensitive data, such as credit card numbers. This issue is effectively addressed by many cryptographic algorithms and tools [Stinson, 95], developed in the past several decades. However, in the presence of active intrudes, which not only listen to the traffic, but are also able to modify existing messages and generate new ones, other dangers also exist. Such an intruder could impersonate one of the protocol participants – for example, he could fool a customer into thinking that he (intruder) was a merchant. The intruder could also modify the contents of the messages, for example to change the shipping address.

Hence, cryptographic algorithms and tools alone are not enough. C, M and B need to agree on a set of rules for message exchange – a protocol, specifically, a payment protocol. Many such protocols have been developed in the past [Bellare, 00; Jarupunphol and Mitchell, 03]. They all try to solve the aforementioned problems with various degrees of success. However, other security issues still exist and here we name three of them. First, M needs to make sure that C is authorized to use the credit card; otherwise M is responsible for "card not present" transaction charge backs. Second, if M has access to the credit card number, C has to trust M that he will not misuse it; what's more, C needs to rely on the security of M's server, since a possible intrusion on the server would reveal the credit card number to attackers. Third, B should not be allowed to see the details of the order; this might be a business secret of M and a privacy concern for C.

Making sure that a protocol addresses all (or some) of these issues is a highly non-trivial task. Security flaws in several systems have been discovered some time after their proposal [Pfitzmann, 95; Kailar, 95; Bella, 05]. In order to make sure that a protocol is secure, one needs to formalize both the protocol and the security requirements that it needs to meet, and then verify that the requirements are met indeed.

Other important problems that protocols face have nothing to do with security. Payment systems implementing the protocols must be simple enough, so that ordinary users (customers) can carry out their purchases. From the customers' point of view, it is also desirable that the payment system allows them to place orders from anywhere. From the merchants' point of view, the integration of the payment system with their existing software solutions must be cost-effective.

In this paper, we propose IPS (Internet Payment System), which we have developed based on lessons learned from existing payment systems. IPS tries to address both classes of issues described above. In section 2, we discuss some of the existing secure payment systems, how those systems work, their advantages and disadvantages. Drawing on conclusions from section 2, we define IPS design goals and list them in section 3. The system is described in detail in section 4. IPS security goals are presented in section 5, while the process of verifying the security of IPS is described in section 6. Before concluding the paper, section 7 presents the comparison between IPS and the systems presented in section 2.

2 Existing Solutions

There have been many proposals for payment systems in the past, both for coin-like systems and check-like systems, which try to imitate cash and paper check payments, respectively. The research on coin-like systems started with the work of Chaum [Chaum, 90a; Chaum, 83], which was deployed commercially in the e-cash payment system by DigiCash. Other well-known systems include Brand's system [Brands, 93], and the system by Okamoto and Ohta [Okamoto and Ohta, 92], which is the first system that offers divisible cash, that is, coins that can be split into several smaller coins after withdrawal.

One of the first cheque-like systems was the one presented in [Chaum, 90b], but many more exist. Among the more important systems was the iKP protocol family [Bellare, 00] designed by IBM in 1995. The iKP family was implemented in a field trial and was a predecessor to the SET payment system. SET [Bella, 00a; Merkow, 98] was developed by Visa and MasterCard (involving other companies such as GTE, IBM, Microsoft and Netscape) starting in 1996. It was heavily publicized in the late 1990's as the credit card approved standard, but it did not attain widespread use. After the failure of SET, Visa developed a new protocol called 3-D Secure [Jarupunphol and Mitchell, 03; Wrona, 01]. Since SET and 3-D Secure were developed by major credit card companies, we devote special attention to them later on in the paper.

In spite of the existence of all these systems, the most widely used way of protecting credit card payments over the Internet today is SSL/TLS [Ford and Baum, 97] – a protocol which was designed to provide security for communication links, not e-commerce transactions per se [Rescorla, 01].

2.1 SSL/TLS

On the Internet, applications usually ensure communication security using the industry standard protocols, namely Secure Sockets Layer (SSL) and Transport Layer Security (TLS). SSL was launched in 1994 by Netscape, with the primary goal of providing secure communications between web browsers and web servers. In 1995, the Internet Engineering Task Force (IETF) introduced a protocol named Transport Layer Security (TLS). While internal differences between these two protocols exist [Rescorla, 01], they provide the same functionality and we refer to them as SSL/TLS.

SSL/TLS secures point-to-point links at the session layer. Its cipher suite includes both asymmetric and symmetric algorithms mechanism. Asymmetric encryption is provided by the means of the RSA (Rivest-Shamir-Adelman) algorithm, DSA (Digital Signature Algorithm) and the Diffie-Hellman key exchange algorithm, while AES (Advanced Encryption Standard), Camellia, DES (Data Encryption Standard), Triple-DES, IDEA (International Data Encryption Algorithm), RC4 (Rivest Cipher 4), and RC2 (Rivest Cipher 2) are used for symmetric encryption.

Although SSL/TLS was not designed to be a payment system, it is certainly possible to use it to accept credit cards payments. In fact, payments by SSL/TLS appear to be the most common way of conducting business on the Internet nowadays [Rescorla, 01; Ford and Baum, 97]. Main reasons for this include:

- Transparence - since SSL/TLS provides security at the session layer, its presence is completely invisible either to the merchants' Web shop software or the customer. This is especially important for merchants because there's no cost for integrating SSL/TLS with their existing systems, other than the cost of installing the certificate.
- Ease of use for customers - SSL/TLS is already built into commonly used Web browsers and there is no need to install any additional software.
- Mobility of customers – they can place an order from any computer equipped with one of the standard Web browsers.
- Low complexity - the system is not complex, resulting in minimal impact on transaction speed.

However, SSL/TLS has some serious problems when it comes to meeting the security challenges of today's financial sector. Because it is based on independent point-to-point connection sessions, SSL/TLS does not support multiple-party or indirect communication very well. In the context of our scenario of interest (see Figure 1), this is reflected in the following shortcomings:

- The merchant cannot reliably identify the cardholder. In cases where customers use stolen credit cards to initiate e-commerce transactions, merchants are responsible for "card not present" transaction charge backs. While SSL/TLS does provide the possibility of client authentication with the use of client certificates, such certificates are not obligatory and are rarely used. Furthermore, even if the client possesses a certificate, it is not necessarily linked with his credit card. This means that the client might not be authorized to use the credit card in question.
- SSL/TLS only protects the communication link between the customer and the merchant. The merchant is allowed to see the payment information. SSL/TLS can neither guarantee that the merchant will not misuse this information, nor can it protect it against intrusions whilst it is stored at the merchant's server.
- Without a third-party server, SSL/TLS cannot provide assurance of non-repudiation (see section 5).
- SSL/TLS indiscriminately encrypts all communication data using the same key strength, which is unnecessary because not all data needs the same level of protection. For example, a credit card number needs stronger encryption than an order item list. Using the same key strength for both creates unnecessary computational overhead.

2.2 Secure Electronic Transaction (SET)

SET is an open standard for protecting the privacy of electronic transactions and ensuring their authenticity [Bella, 00a; Merkow, 98]. Unlike SSL/TLS, SET was designed as a payment system. SET transactions include the following participants: the customer, the merchant and the payment gateway. SET provides authentication of each party in a SET transaction by the use of digital certificates. These certificates are issued by a trusted third party known as a Certification Authority (CA), which vouches for the identity of the certificate holder. This means that even the customer needs to register with a CA before he may engage in transactions.

The SET protocol relies on two different encryption mechanisms, as well as an authentication mechanism. SET uses symmetric encryption, in the form of the DES algorithm, as well as asymmetric, or public-key, encryption, in the form of the RSA algorithm.

SET strong points are as follows:

- It fulfills the following fundamental security requirements (see section 5): confidentiality, authentication and data integrity. This was verified by a large collection of security proofs based on formal methods [Bella, 05; Bella, 00b; Bella, 02]
- In the standard variant of the protocol, SET prevents merchants from seeing the customer payment information, since this information is encrypted using the payment gateway's public key.
- To ensure merchant privacy, SET prevents the payment gateway from seeing the order information.

Despite these advantages, SET failed to win significant market adoption. The main reason for this was the complexity of use for the customers. This is reflected in the following:

- The customer must install additional software, which can handle SET transactions.
- The customer must have a valid digital certificate.

As a consequence, the customer is also not allowed to place an order from PCs other than his/her SET-initialized PC.

Some other disadvantages of SET are as follows:

- Implementing SET is more costly than SSL/TLS for merchants as well. Adapting their systems to work with SET is more complicated than adapting them to work with SSL/TLS. Furthermore, merchants must have accounts opened at business banks capable of handling SET transactions.
- Business banks must hire companies to manage their payment gateways, or install payment gateways by themselves.
- Despite being designed with security in mind, SET also has some security issues. In a variant of the SET protocol, the merchant is allowed to see the customer payment information [Fritscher and Kump, 00], just as with SSL/TLS. There are also some other, minor security issues in this protocol [Bella, 05].
- SET employs complex cryptographic mechanisms that may have an impact on the transaction speed.

2.3 3-D SECURE

3-D Secure is built upon the relationships between three domains, named the acquirer, the issuer, and interoperability domains [Jarupunphol and Mitchell, 03; Wrona, 01]. The acquirer domain covers the relationship between the merchant and the acquirer.

The issuer domain covers the relationship between the cardholder and the issuer. The interoperability domain supports the relationship between the acquirer and issuer domains.

In 3-D Secure, the payment gateway, which provides an interface between the merchant/acquirer's payment system and the Visa proprietary payment network VisaNet, must be implemented in the acquirer domain [Jarupunphol and Mitchell, 03]. Merchants are responsible for installing an SSL/TLS Merchant Plug-In (MPI) at their servers, as would normally be the case if they wish to implement SSL/TLS for customer-merchant communication protection. Within the issuer domain, each card issuer is required to maintain a special server known as the Access Control Server (ACS). The ACS is used to support cardholder authentication. The Visa directory is a server in the interoperability domain, used to enable communication between merchant servers and card issuers.

To protect the security of communication between the various entities, 3-D Secure requires the following links to be protected using SSL/TLS: cardholder-merchant, cardholder-ACS, merchant-Visa Directory, and Visa Directory-ACS.

Since 3-D Secure bases its protection on SSL/TLS connections, it shares the advantages of SSL/TLS, in terms of ease of use and mobility of customers. Unfortunately, it also shares some of the disadvantages. The merchant still has access to the payment information, and all information is encrypted using the same key strength. The main advantage over SSL/TLS is that 3-D Secure provides credit card authorization and non-repudiation. On the other hand, prior customer registration is required.

3 Design Goals

Based on the advantages and disadvantages of existing payment systems identified in Section 2, we defined the objectives for IPS. These objectives are based on security requirements for an efficient payment system, as well as end-user requirements. Namely, objectives of IPS are to create:

- an easy to use, low-cost system for secure credit and debit card transactions between customers, merchants, and banks;
- a system in which customers are not required to undergo any form of registration prior to engaging in IPS transactions. Specifically, this means that:
    - customers are not required to install additional software for secure payments
    - customers are not required to have a digital certificate;
- a system that will satisfy the fundamental requirements to be considered secure (see section 5);
- a system that uses strong cryptography and authenticity checking models;
- a system that prevents the merchant from seeing payment information;
- a system that prevents card issuers from seeing order information, protecting merchants’ privacy.

Based on the criteria above, we have developed the IPS system that is presented in the next section.

4 The Proposed System: Internet Payment System

This section gives a full description of IPS. IPS consists of five system segments (participants). The communication between the participants goes through two phases: the preparation phase and the IPS protocol phase.

The system description starts with the definition of the system's principal entities (subsection 4.1). We then turn out attention to the communication between the participants. The preparation phase is described in subsection 4.2, followed by the description of the IPS protocol itself (subsection 4.3). Subsection 4.4 describes the certificate architecture of the system. An implementation of IPS is presented in subsection 4.5.

4.1 Participants

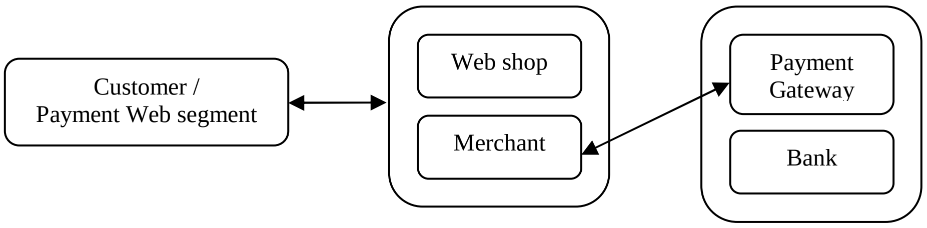

Figure 2: The principal participants in IPS

Compared to the typical e-commerce scenario (Figure 1), IPS adds additional participants to the system. The principal participants in IPS are the customer (the payment Web segment), the merchant, the merchant Web shop, the payment gateway, and the bank (Figure 2). The "merchant" is a piece of software running on the merchant's server. For the time being, we will neglect the details of the connection between the merchant and the merchant Web shop and view them as being the same. The payment Web segment is a digitally signed piece of software running on the customer's computer (typically a digitally signed Java applet), automatically downloaded from the merchant server prior to the protocol execution. It handles all the protocol interaction on the customer's side, so for the sake of simplicity we will use the term "customer" interchangeably with "payment Web segment". It should be clear from the context which one of the two we are referring to.

4.2 The Preparation Phase

With every payment system, it is necessary to conduct certain preparation steps (at the very least, choosing the goods) before the payment itself can take place. Specifically, for IPS, the preparation phase consists of two sub-phases, the ordering phase and the payment Web segment download phase.

During the ordering phase, the communication between the customer and the Web shop (Figure 2) takes place through standard protocols, HTTP or HTTPS. The customer browses the merchant's Web shop and decides which goods to buy.

When the customer presses the "pay" button, the payment Web segment is downloaded from the merchant server. This is done through HTTPS. The payment Web segment is a piece of software, digitally signed by a trusted party (the payment gateway) to prevent any possible modifications to it by the merchant.

4.3 Protocol Description

Before introducing the IPS protocol, we define the notation used in the description. We use the standard Alice-Bob notation, in which the transmitter and the receiver are noted first, followed by the contents of the message. The notation is summarized in Table 1.

Notation	**Description**
$\{m\}_X$	(Hybrid) encryption of the message *m* with the key *X*.
`PubKX`	Party *X*'s public key.
`SigX(m)`	Signature of the message *m* by party *X*.
`X -> Y:   Z`	Message *Z* sent from party *X* to party *Y*.

Table 1: Notation used in the protocol description.

Hybrid encryption of a message means that the message is encrypted using a symmetric session key, which is in turn encrypted using an asymmetric key. The session key is then used for the remainder of the session between the two parties. For example, the message:

C -> M: $(\{\text{PubKC}\}_{\text{PubKM}})$

actually means:

C -> M: $(\{\text{Kcm}\}_{\text{PubKM}}, \{\text{PubKC}\}_{\text{Kcm}})$

and Kcm is used for the encryption of all further messages between C and M (in the place of PubKC and PubKM).

In the protocol description we use a single public/private key pair for both signatures and encryption. An actual implementation might opt to use two different pairs of keys for signatures and encryption (this in fact might even be necessary, depending on the encryption/signature scheme used).

One of the main IPS design goals was to create a system in which the customers will not be required to perform any kind of registration before they may engage in IPS transactions. However, prior registration still remains an option. In IPS, prior registration means obtaining a digital certificate from an IPS CA (see subsection 4.4).

Thus, there are two scenarios of the protocol. Which scenario will take place is determined by the payment Web segment, which searches the customer's computer for a valid customer certificate. In the first scenario, the customer does not have a digital certificate issued by an IPS CA

4.3.1 First Scenario – No Customer Certificate

In this scenario, the payment Web segment first prompts the customer for the relevant payment information. This information includes a challenge code, which will authenticate and authorize the customer to use that credit card. For this purpose, any sort of secret code issued by the bank will do; for example, the credit card PIN code can be used. In the rest of the paper, we refer to this code simply as PIN.

Additionally, the payment Web segment creates a public/private key pair. Based on this pair, the payment Web segment then creates a self-signed customer digital certificate. Note that this is not a certificate in the usual sense of the word; it is not signed by a CA and at this stage it does not provide any assurance of the identity of its owner. The ownership of the certificate can only be established later, as described at the end of this section. After the ownership is established, the certificate can later be used in dispute resolution in a similar manner to the certificates issued by a CA, which is why will refer to it as a "certificate". This certificate is temporary and will be used for this session only. The certificate is not installed on the user's computer. It is simply sent to the merchant and the payment gateway as described later in this subsection. The user is not required to interact with this process in any way; as all of this is done in background.

Now the protocol starts to execute. For clarity, the general outline of the protocol is given in Figure 3. C stands for the customer, M for the merchant and PG for the payment gateway from Figure 2.

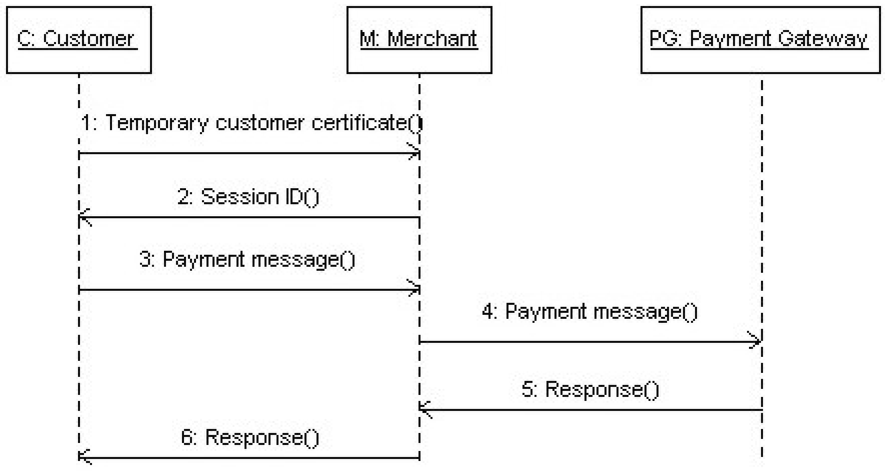

Figure 3: Message exchange in the IPS protocol

Now we describe each step from Figure 3 in detail, using Alice-Bob notation (Figure 4).

```
1. C -> M: ({PubKC}PubKM)
2. M -> C:
   ({Sid, SigM(Sid)}PubKC)
3. Let:
   PInf = (CardInformation, Amount, PIN, Sid, M)
   PM = {PInf, Non_CPG, SigC(PInf, Non_CPG)}PubKPG
   OrderInf = (OrderDescription, Amount, Sid)
   OrderSignature = SigC(OrderInf)
   C -> M:  {PM, OrderSignature}PubKM
4. M -> PG:
    ({PM, PubKC, SigM(Amount, PubKC, Sid)}PubKPG)
5. PG -> M:
    {Response, SigPG(Response, Sid, Amount, Non_CPG)}PubKM
6. M -> C:
    {Response, SigPG(Response, Sid, Amount, Non_CPG)}PubKC
```

Figure 4: Alice-Bob notation of the IPS protocol

In the beginning of the protocol, the customer transmits the temporary customer certificate to the merchant (Figure 4, message 1):

$\texttt{C -> M: (\{PubKC\}}_{\texttt{PubKM}}\texttt{)}$

The certificate is encrypted with the merchant's public key (recall that we use hybrid encryption, section 4.3). The merchant responds with (Figure 4, message 2):

$\texttt{(\{Sid, SigM(Sid)\}}_{\texttt{PubKC}}\texttt{)}$

Here *Sid* is a freshly generated number, which serves as an ID for the session. It is used to identify the transaction, and should ideally be unique for each session. In practice, this means that it should really be a fairly large number. *Sid* and its digital signature (to confirm the merchant's identity) are encrypted using the customer's public key.

The next step (creating message 3 from Figure 4) is a little more complicated. Based on the user input, the payment Web segment generates the relevant payment information, including the session ID and the merchant's name *M*:

```
PInf = (CardInformation, Amount, PIN, Sid, M)
```

Then the payment message is prepared:

$\texttt{PM = \{PInf, Non\_CPG, SigC(PInf, Non\_CPG)\}}_{\texttt{PubKPG}}$

It consists of the payment information, a fresh nonce for the payment gateway and the digital signature of the two. The whole message is encrypted using the payment gateway's public key. This prevents the merchant from seeing the credit card information. Now the customer also gathers the relevant order information and digitally signs it.

```
OrderInf = (OrderDescription, Amount, Sid)
OrderSignature = SigC(OrderInf)
```

The merchant can generate *OrderInf* as well (since he knows the order description, the amount and *Sid*); he can then make sure that the customer signed the same order information. If this is the case, he then stores this digital signature along with the order information itself. This can later be used as a proof to verify what the customer has ordered, as will be described in section 6.3.

Now the merchant starts talking to the payment gateway (Figure 4, message 4):

$(\{\texttt{PM, PubKC, SigM(Amount, PubKC, Sid)}\}_{\texttt{PubKPG}})$

The payment message *PM* is forwarded to the payment gateway, since the merchant cannot decrypt it. The merchant also includes the customer's certificate in the message. The customer's certificate, the amount and the session ID are digitally signed to confirm the merchant's identity.

Upon receiving the payment message, the payment gateway decrypts it using its secret key. It checks the signature of the payment information, and checks if the customer is authorized to use the credit card in question. The authorization is performed based on the combination of the credit card number and the PIN. If the combination is valid, the customer is authorized to use the credit card. This also means that the ownership of the temporary certificate is established – the owner of the certificate is the owner of the credit card. This is very important because of possible disputes, as will be discussed in section 6. A check is also made to make sure that both the merchant and the customer agree on the amount to be charged. Finally, the payment gateway also checks whether the combination of *Sid* and *Non_CPG* is fresh, to eliminate possible replay attacks by malicious merchants. If everything checks out, the credit card information and the amount are forwarded to the bank. The bank checks whether there is enough money on the account. If so, the transfer is made to the merchant's account and the bank returns a positive response; otherwise a negative response is returned. If the response is positive, the payment gateway stores the details of the transaction (including the customer's temporary certificate) in its database.

Then the payment gateway forwards the response to the merchant (Figure 4, message 5):

$\{\texttt{Response, SigPG(Response, Sid, Amount, Non\_CPG)}\}_{\texttt{PubKM}}$

The response, along with other information (namely, *Non_CPG*) is digitally signed so the customer can verify that the response came from the payment gateway and that it is related to the current transaction (i.e. that it is not replayed by the merchant). The merchant forwards the contents of the message to the customer (Figure 4, message 6):

$\{\texttt{Response, SigPG(Response, Sid, Amount, Non\_CPG)}\}_{\texttt{PubKC}}$

4.3.2 Second Protocol Scenario

In the second protocol scenario, when the customer has a valid digital certificate, the protocol is somewhat simpler. In this scenario, we assume that all certificates, including the customer's certificate, are already exchanged so there is no need to send it in step 1 of the protocol (Figure 3). Also, the payment information in protocol step 3 (Figure 3) does not contain the PIN. Instead, the authorization of the cardholder is performed based on his certificate. The rest of the protocol remains the same.

4.4 Certificate Architecture

There are authentication steps between participants in every phase of the protocol. This is important to prevent an unknown third party from inserting itself between IPS protocol participants. To achieve this, IPS uses a hierarchy of trust, as presented in Figure 5. IPS CAs are arranged in a hierarchical structure with the IPS Root CA as the root level Certificate Authority and the Customer, Merchant, Payment Gateway and Bank Certificate Authorities as peer CAs at the next level. Certificates are used for message encryption, digital signatures and digital envelope creation.

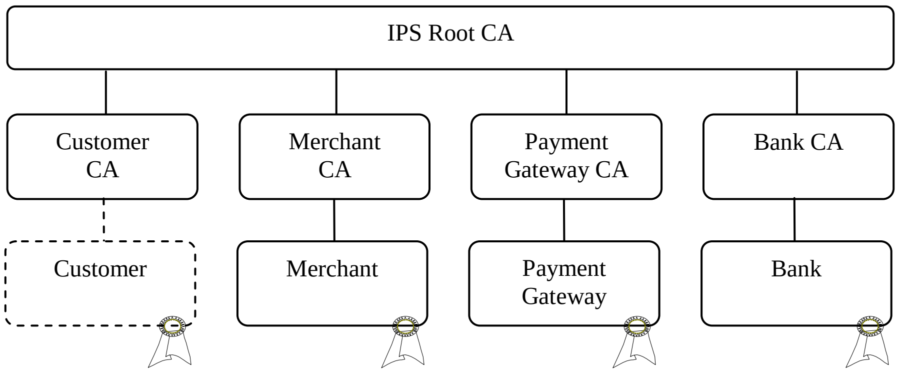

Figure 5: Hierarchy of trust. All parties hold cerificates signed by corresponding CAs.

This certificate architecture consists of eight obligatory components and one optional component. As specified in the design goals in section 3, the customer is not required to have a digital certificate, so the customer component is optional. In that case, the payment Web segment creates a temporary certificate as described in section 4.3.

4.5 An Implementation of IPS

We have made an experimental implementation of IPS for academic purposes. Platform independence was ensured by using Java technology. Java 2 Standard Development Kit 5.0 was used for system development.

The payment Web segment was developed as a digitally signed Java applet. Since Java support is already built into standard Web browsers, the customer does not need to install any additional software. Hence, the payment Web segment, and consequently IPS can be used from any computer, achieving some of the main design goals: ease of use and mobility of the customers.

Merchant and payment gateway servers were implemented as Java server applications. It is easy to integrate the developed merchant server with existing Web shop solutions, regardless of the programming language used for their development. The merchant's Web shop needs to do two additional things. It needs to pass the order description to the payment Web segment, and implement a Web service which receives the notification from the merchant whether the purchase was approved or not.

In our implementation we used the RSA algorithm for asymmetric encryption, and IDEA for symmetric encryption. Using RSA enabled us to use one pair of keys for both encryption and signatures. In addition, we developed the IPS CA architecture based on the publicly available OpenSSL library.

5 Security Requirements

As specified in the design goals, security is paramount to the IPS system. It is generally accepted that, in order to be considered secure, a payment system must satisfy the following fundamental security requirements [Lawrence, 98; Ford and Baum, 97; Stallings, 95; Merkow, 98; Bhimani, 96]: confidentiality (secrecy), data integrity, authentication and non-repudiation. In this section, we discuss each of these requirements and how they relate to IPS.

Confidentiality (secrecy) – preventing the disclosure of data to unauthorized parties. All communication between parties is restricted to the parties involved in the transaction. Confidentiality is an essential component in user privacy, as well as in the protection of proprietary information, and as a deterrent to theft of information services. The only way to ensure confidentiality on a public network is through strong encryption. We demand that IPS guarantees the secrecy of the credit card information between the customer and the payment gateway. Furthermore, when the shopping process preceding the execution of IPS itself takes place through HTTPS, we demand that IPS guarantees the secrecy of the order description between the customer and the merchant (when the shopping process takes place through HTTP, an attacker can learn the order details beforehand).

Authentication – ensures that the parties are who they claim to be, i.e. prevents the masquerade of one of the parties involved in the transaction. Both parties should be able to feel comfortable that they are communicating with the party with whom they think they are communicating. Applications usually perform authentication checks through security tokens or by verifying digital certificates issued by certificate authorities. We demand that IPS guarantees mutual authentication between all pairs of participants (i.e., customer – merchant, merchant – payment gateway and customer – payment gateway).

Data integrity – verifies that the communication has not been altered in transmission, i.e. prevents unauthorized modification of data. Similarly, it should not be possible to modify data while in storage. Financial messages travel through multiple routers on the open network to reach their destinations, so we must make sure that the information is not modified in transit. Applications usually assure data integrity using digital signatures. We demand that IPS guarantees the following: integrity of the order information between the customer and the merchant, integrity of the amount, the session ID and the response between the merchant and the payment gateway, and the integrity of the payment information and the response between the customer and the payment gateway.

Non-repudiation – is concerned with ensuring that each participant in the protocol is provided with convincing evidence of the other's participation in a protocol session. Neither party should be able to deny having participated in a transaction after it took place. Non-repudiation is usually provided through digital signatures and public-key certificates. We demand that IPS guarantees that neither the merchant nor the customer can deny their participation in the protocol.

6 Security Verification Process

Having specified the main security requirements for IPS in section 5, we now turn on to the process of verifying those requirements.

Establishing the security of a protocol can be a tricky issue. Subtle flaws in protocol design can be very hard to detect, but can cause various security goals to be compromised. A famous example is the Needham-Schroeder authentication protocol in which a flaw was found 18 years after the protocol was first published [Needham and Schroeder, 78; Lowe, 96]. Hence, formal verification of protocol correctness is desirable.

Cryptographic proofs deal with probabilities and computational issues, and reduce the issue of protocol security to the security of its cryptographic primitives. However, these proofs quickly become very complex even for the simplest of protocols. Thus, most proofs of protocol security abstract the computation, and axiomatize the cryptographic primitives. This means that they assume perfect cryptography (i.e., the only way for someone to read an encrypted message is to obtain possession of the corresponding key). Virtually all of those proofs use the so-called Dolev-Yao models [Dolev and Yao, 83]. Recent results [Abadi and Rogaway, 02; Micciancio and Warinschi, 04] partially confirm the computational soundness of such abstractions.

Still, even with the cryptographic details out of the way, the protocol security problem is anything but trivial (and is, in fact, undecidable in general as shown in [Durgin, 99]). Distributed aspects of protocols cause the number of cases to be considered to grow at a very fast rate. Hence, hand-written proofs are hard to carry out with rigor, and thus tend to get sketchy and, as a consequence, error prone [Pfitzmann, 95]. Thus, (semi-) automation of proofs is generally desirable. Several such methods have been devised over the years. Assuming fixed bounds on the protocol message size and the number of concurrent protocol runs enables one to effectively model a protocol as a finite-state system, and then to apply finite-state model-checkers to perform an analysis of the protocol [Mitchell, 97]. One can also model attacks on a protocol as symbolic constraints, and then use constraint-solvers to analyze the protocol; this approach does not assume bounds on the protocol message size [Millen and Shmatikov, 01]. Similarly, combining model-checking with the so-called lazy datatypes enables consideration of infinite state-spaces and removes the bounds on the message size [Basin, 05]. Other approaches involve examining possible protocol traces, i.e. sequences of events in a protocol, and then inductively proving (with the help of a theorem-prover) that protocol security properties hold in every trace [Paulson, 98]; this proves the security of the protocol over an unbounded number of protocol runs, but requires a lot of human interaction, and undecidability results have to be kept in mind.

For the purpose of verifying the security of IPS, we have opted to use a recently proposed tool, called AVISPA [Armando, 05], which stands for Automated Verification of Internet Security Protocols and Applications. We chose to use AVISPA for two reasons:

1. It is easy to use for protocol modelers. This enabled us to focus on the protocol model itself;
2. The same protocol model can be checked with several tools which use different approaches.

6.1 AVISPA and the model of IPS

AVISPA enables one to specify a protocol via a so-called High-Level Protocol Specification Language, HLPSL [Von Oheimb, 05]. HLPSL specifications are role-based, and can readily be generated from Alice-Bob notations. The protocol modeler creates a role for each party in the protocol, modeling protocol steps as transitions in a role, and then models the desired security properties of the protocol. The specification is then translated into a description of an infinite-state transition system, which is fed to several back-end tools for completely automated analysis. These tools search the state space for states that violate the specified security properties. The back-ends employ the standard Dolev-Yao model, in which the intruder is assumed to have complete control over the network. He can intercept all messages from all parties, and he can generate new ones (based on his current knowledge), and send them under any party’s name. However, the perfect cryptography assumption holds.

For the purpose of modeling the IPS protocol, we have performed several simplifications, as follows. We have abstracted the details of the payment gateway – bank connection, which is considered to be secure, and modeled only the roles of the customer, the merchant and the payment gateway. We have also assumed that the shopping process itself already took place, i.e. that both the customer and the merchant know the order details and the amount beforehand.

We have modeled both scenarios (from section 4.3) of the IPS protocol. HLPSL specification of the customer role for the first IPS scenario (in which a new temporary certificate is generated for the customer for each session) is given in Figure 6.

```
role cardholder(C,M,P: agent,
                CardInf: text,
                Amount : nat,
                OrderDesc : text,
                PubK_M,
                PubK_PG : public_key,
                PIN : message,
                Hash : hash_func
               ) played_by C def=
 local State : nat,
       Non_CPG, Sid, Response : text,
       OI, PI : message,
       Kcm : symmetric_key,
       PubK_C : public_key,
       SND, RCV: channel (dy)
 init State := 0
 transition
 1. State = 0 /\ RCV(start)
    =|>
    State' := 2 /\ PubK_C' := new()
                /\ Kcm' := new()
                /\ SND({Kcm'}_PubK_M.{PubK_C'}_Kcm')
 2. State = 2   /\ RCV({Sid'.{Hash(Sid')}_inv(PubK_M)}_Kcm)
    =|>
    State' := 4 /\ Non_CPG' := new()
                /\ PI' := CardInf.Amount.Sid'.Non_CPG'.PIN.M
                /\ OI' := OrderDesc.Amount
                /\ SND({{PI'.{Hash(PI')}_inv(PubK_C)}_PubK_PG.
                   {Hash(OI'.Sid')}_inv(PubK_C)}_Kcm)
                /\ secret(OrderDesc,order,{C,M})
                /\ secret(Amount,order,{C,M,P})
                /\ secret(CardInf,payment,{C,P})
                /\ witness(C,M,cm_deal,Sid'.OI')
                /\ witness(C,P,cp_deal,PI')
 3. State = 4   /\ RCV({Response'.{Hash(Response'.Sid.
                   Amount.Non_CPG)}_inv(PubK_PG)}_Kcm)
    =|>
    State' := 11 /\ request(C,M,mc_deal,Sid.OI)
                /\ request(C,P,pc_deal,PI)
                /\ request(C,P,pc_response,Response')
end role
```

Figure 6: Model of the customer in the first scenario of IPS protocol

We will not describe the HLPSL semantics in detail (for more information on HLPSL look in [Von Oheimb, 05]); we will just point out some elements important for our model. As shown in Figure 6, the description of the customer role begins with a list of role parameters such as the credit card information (*CardInf*), the amount (*Amount*), the order description (*OrderDesc*), etc. Next is the list of variables local to the role (keyword *local*) such as the nonces (*Non_CPG*) and the session ID (*Sid*). The key parts of the role specification are the transitions. Our customer role has three possible transitions. A transition is fired when the role instance is in an appropriate state (as specified by the *State* variable in our model) and upon receipt of an adequate message. The first transition corresponds to the protocol message 1 (message numbers from this paragraph refer to the Figure 4). In this transition, a fresh session key and a temporary certificate are generated using the HLPSL function *new()*. The message is sent to the merchant over a public channel. The trigger for the second transition is the protocol message 2. Note that the digital signatures are represented as encryptions of message hashes using the corresponding private keys, as this is the only way to represent digital signatures in AVISPA. The result of the transition is the protocol message 3 and a series of *secret* and *witness* events that will be described in the subsequent paragraph. The third transition corresponds to the protocol message 6, in which the response from the payment gateway is received and the protocol is terminated. In this transition we generate the *request* events, also described in the subsequent paragraph. The merchant and the payment gateway roles are modeled in a similar fashion. The merchant role transitions correspond to the protocol messages 1-6, while the payment gateway role transitions correspond to the protocol messages 4 and 5.

As stated in section 5, we demand that the protocol meets several security goals. These demands are modeled as events in HLPSL. When a party transmits a message that it wishes to be kept secret between the parties in a certain set, it generates a *secret(m, id, {a, b})* event. This means that the message *m* should remain secret between the parties in the set (here *a* and *b*), creating a security goal identified by *id*. If there is a state in the transition system where the intruder learns *m* when he does not belong to the specified set, the goal *id* is violated. As can be seen from the customer role and its *secret* events, we demand that the credit card information remains secret between the customer and the payment gateway (a goal identified as *payment*). We model similar demands for the amount and the order information.

Authentication goals are achieved through *witness(a, b, id, m)* and their corresponding *request(b, a, id, m)* events. A *witness* event means that the party *a* asserts that it wishes to communicate with *b*, using the value *m*, for the purpose identified by *id*. A *request* event means that *b* now believes that *a* agrees with it on the value of *m*, for the goal *id*. A weak authentication goal *id* is violated whenever a state is reached where a *request(b, a, id, m)* event is generated, for which there was no previous *witness(a, b, id, m)* event. Strong authentication additionally requires that *request* events do not occur more times than the corresponding *witness* events, meaning that *b* requires *a* to be alive, and thus eliminating replay attacks. An example of those events can be seen in the customer role. For example, in the second transition, the customer acknowledges that he wishes to communicate with the party M, using certain values (the session ID and the order information), for the purpose of *cm_deal*. Later, in the third transition, the customer requests that a similar guarantee be produced for him by M during the protocol execution, for the purpose of *mc_deal*. Note that checking authentication in this way implicitly ensures data integrity, since the values in the *request* events must match those stated in the *witness* events.

HLPSL requires that a special role, called session, be specified. This role corresponds to a single session of the protocol. It can be basic (consisting of just one role) or composed (consisting of multiple roles). In the case of IPS, it is defined as a composition of customer, merchant and payment gateway roles (Figure 7).

```
role session(C,M,P: agent,
                CardInf : text,
                Amount : nat,
                OrderDesc : text,
                PubK_M,
                PubK_PG : public_key,
                PIN : message,
                Pin_func : hash_func,
                Hash : hash_func
                ) def=
composition
 customer(C,M,P,CardInf,Amount,OrderDesc,PubK_M,PubK_PG,PIN,Hash)/\
 merchant(C,M,P,Amount,OrderDesc,PubK_M,PubK_PG,Hash)/\
 paymentgateway(C,M,P,PubK_M,PubK_PG,Pin_func,Hash)
end role
```

Figure 7: Model of an IPS session in HLPSL

In the analysis, we modeled three parallel sessions: one with honest participants, one with a malicious customer, and one with a malicious merchant. In HLPSL this is achieved through a top-level role usually called environment. The environment role for the first IPS scenario is given in Figure 8. As can be seen from the figure, we assumed the payment gateway to be honest (its role is never played by the intruder *i*). We hypothesize that this is a reasonable assumption, since some assumptions about trusted parties must be made (e.g. about certificate authorities); a dishonest payment gateway is allowed to view sensitive credit card information anyway, and poses a major security problem.

```
role environment() def=
  const h : hash_func,
  pin_func : hash_func,
  order,payment : protocol_id,
  mp_deal,pm_deal,cm_deal,mc_deal,cp_deal,pc_deal : protocol_id,
  pm_response,pc_response : protocol_id,
  c,m,p : agent,
  pubk_m,pubk_pg,pubk_i : public_key,
  cardInf_c,cardInf_i,orderDesc1,orderDesc2,orderDesc3 : text,
  amount1,amount2,amount3 : nat
 intruder_knowledge = {c,m,p,pubk_m,pubk_pg,pubk_i,inv(pubk_i),
   cardInf_i,amount2,orderDesc2,amount3,orderDesc3,h,
   pin_func(i.cardInf_i)}

composition

 session(c,m,p,cardInf_c,amount1,orderDesc1,pubk_m,pubk_pg,
   pin_func(c.cardInf_c),pin_func,h) /\
 session(i,m,p,cardInf_i,amount2,orderDesc2,pubk_m,pubk_pg,
   pin_func(i.cardInf_i),pin_func,h) /\
 session(c,i,p,cardInf_c,amount3,orderDesc3,pubk_i,pubk_pg,
   pin_func(c.cardInf_c),pin_func,h)
end role
```

Figure 8: Environment role of the IPS model in HLPSL

The *intruder_knowledge* set from Figure 8 contains all the constants that the intruder knows prior to the protocol execution. Note that in this model, we exclude the constants *amount1* and *orderDesc1* from that set, that is, we assume that the shopping process takes place through HTTPS. The payment gateway is equipped with a function called *pin_func* which we use to model credit card authorization based on credit card number/PIN combinations; this function enables the payment gateway to create a map between credit card numbers, PINs and user identities.

6.2 Verification results

Based on the IPS security requirements from section 5, we modeled the following goals: secrecy of the order description between the customer and the merchant; secrecy of the credit card information between the customer and the payment gateway; and (strong) mutual authentication between all pairs of participants (i.e., customer – merchant on the order information, merchant – payment gateway on the amount and the response, and customer – payment gateway on the payment information and the response). If the shopping process takes place through HTTPS (i.e., the attacker does not know the order details beforehand), we can also demand the secrecy of the order description between the customer and the merchant.

As noted before, strong authentication between two parties on a certain property implicitly ensures the data integrity of such a property between those two parties. Currently, non-repudiation goals cannot be modeled explicitly in AVISPA [Santiago and Vigneron, 05]; we discuss non-repudiation in section 6.3.

We ran the analysis of the protocol in two variants. In the first one, we assumed a strongly-typed model, i.e., a model in which a message field is always interpreted according to its type, and no two fields of different types can be substituted one for the other. Two of the back-end AVISPA tools, CL-AtSe (a constraint solver) and OFMC (a symbolic model checker [Basin, 05]) have verified the protocol to be safe under the assumption of a bounded number of sessions. In other words, they found that the protocol achieves all the modeled goals. The analysis was apparently too complex for the current versions of the other two tools (SATMC [Armando and Compagna, 04], a SAT-based model checker, and TA4SP [Boichut, 05], a tree automata based tool), since we were unable to get any output from them in a reasonable amount of time. We hope that newer versions of those tools will be able to analyze our protocol, TA4SP in particular, since it attempts to prove secrecy over an unbounded number of sessions.

In the second variant, we assumed an untyped model. Both CL-AtSe and OFMC have discovered several type-flaw attacks that compromise the authentication goals. While prevention of type-flaw attacks can easily be achieved during (and is probably best left to) the implementation of the protocol, as suggested in [Heather, 00], we have also devised another, slightly modified version of the protocol which is not vulnerable to such attacks, as confirmed by both CL-AtSe and OFMC. However, the price paid is the addition of some extra hashes, encrypted fields etc. which add unnecessary computational overhead.

6.3 Further notes

As mentioned earlier, non-repudiation goals as such cannot be modeled in AVISPA [Santiago and Vigneron, 05]. However, it is well known that non-repudiation is a form of authentication [Ryan, 00]. Since the payment gateway authenticates both the customer (on the payment information) and the merchant (on the amount), neither of them can repudiate their participation in the transaction. In the case of a dispute, the corresponding entry in the payment gateway database can be used as a proof of non-repudiation.

Another dispute that can arise is the one concerning the contents of the customer's order. In this case, the burden of proof lies on the merchant. If the merchant fails to provide the proof, the customer wins the dispute. We contend that the digital signature of the order information, `SigC(OrderInf)`, (Figure 4, message 2) combined with the corresponding entry in the payment gateway database (containing the user certificate), provides the merchant with sufficient evidence of what the customer has ordered. First, note that the merchant cannot generate this message by himself because he does not know `C`'s private key. Hence, it must originate from the network, so this can be expressed as an authentication problem on `SigC(OrderInf)`. Since authentication on that particular message is reached (as discussed earlier) it can be claimed that the message is received from the customer and not from someone else.

7 Comparison With Existing Solutions

Having presented our system in the previous sections, we now give a comparison of IPS with other relevant solutions described in section 2. Comparing IPS with them, we find that the main advantages of IPS are:

- It uses strong cryptography and authenticity checking models (like SET). Through these it achieves privacy, data integrity, authentication and non-repudiation.
- It uses encryption selectively, based on the sensitiveness of the data. For example, payment information is encrypted very strongly (using asymmetric encryption).
- The merchant is prevented from seeing credit card information (unlike SSL/TLS or 3-D Secure, for example). This removes the possibility of leaking this information through intrusions on merchants' servers.
- The payment gateway needs only to communicate with certified merchants, reducing its exposure to the outside world.
- The customer has to be authorized to use his/her credit card before the payment takes place (unlike SSL/TLS).
- The system enables the customer to place an order from anywhere, achieving the same transparency as SSL/TLS:
    - The customer is not required to have a digital certificate (unlike SET),
    - The customer is not required to install additional software for secure payments (unlike SET), even if he/she has a digital certificate
    - The customer does not need to perform an enrollment process beforehand (unlike 3-D Secure and SET).

Table 2 summarizes the results of this comparison.

Property	Protocol			
	SSL/TLS	3D SECURE	SET	IPS
No additional software required	YES	YES	NO	YES
No customer certificate required	YES	YES	NO	YES
Credit card authorization	NO	YES	YES	YES
Merchant cannot see payment information	NO	NO	YES	YES
Strong encryption of sensitive information	NO	NO	YES	YES
No trusted third party required	YES	NO	NO	NO
No customer preliminary registration required	YES	NO	NO	YES
Non-repudiation	NO	YES	YES	YES

Table 2: A comparison of IPS with SSL/TLS, SET and 3-D SECURE

Of course, there are also some drawbacks in using IPS. Many of these drawbacks are shared with other solutions (such as SET and 3-D Secure):

- Merchants need to adapt their existing Web shops to work with IPS. Still, this process is not very complicated, as we discussed in the implementation section (section 4.5)
- Merchants must have accounts opened at banks capable of handling IPS transactions
- Business banks must hire companies to manage their payment gateways, or install payment gateways by themselves.

8 Conclusion

This paper has examined a typical e-commerce scenario, in which customers use credit cards to buy physical goods. This scenario gives rise to different classes or problems, some having to do with security, and some having to do with end-user requirements.

First, we have evaluated state-of-the-art solutions, and then presented a payment system called IPS. IPS is designed with the aim of resolving some of the deficiencies of the evaluated solutions. Main advantages of IPS are: it uses strong cryptography and authenticity checking models; the merchant is prevented from seeing payment information; the system is easy to use for the customer, since he is not required to install additional software for secure payments or to have a digital certificate.

Electronic payment systems are especially sensitive to security issues; the failure of such systems results in direct loss of money and for this reason is highly attractive for intruders. The only way to protect sensitive financial information when transmitted over public networks is to use cryptographic algorithms and techniques, but in the presence of active intruders cryptographic techniques alone are not enough. To ensure that a payment protocol satisfies fundamental security requirements these techniques must be carefully combined. Today there is common consent that payment protocols need a proof of security to be acceptable.

However, with most protocols, formal security verification is an afterthought. In contrast, formal methods have been included in the IPS design process from the very beginning. Having gone through this process, we are now even more assured of the importance of formal verification. Small omissions which creep into complex protocols and which can compromise them severely can be nearly impossible to detect without formal verification.

To prove the security of IPS, we modeled the protocol using HLPSL. We have run the analysis using the AVISPA tool. The results of the analysis were positive; in other words the protocol achieves the desired properties: confidentiality, integrity, authentication and non-repudiation.

Currently, we are working on a modification of the system, trying to optimize it for purchase and delivery of electronic goods, while retaining the achieved properties of IPS.

References

[Stinson, 95] D. Stinson, Cryptography: Theory and Practice, CRC Press, Boca Raton, Florida, 1995.

[Rescorla, 01] E. Rescorla, SSL and TLS — Designing and Building Secure Systems, Addison-Wesley, Massachusetts, 2001.

[Pfitzmann, 95] B. Pfitzmann, M. Schunter, M. Waidner, How to break another “provably secure” payment system, Advances in Cryptology: EUROCRYPT ’95, 1995, 121–132.

[Kailar, 95] R. Kailar, Reasoning About Accountability in Protocols for Electronic Commerce. In Proceedings of the 14th IEEE Symposium on Security and Privacy, 1995, 236-250.

[Bella, 05] G. Bella, F. Massacci, L.C. Paulson, An overview of the verification of SET. International Journal of Information Security, 2005, 17-28.

[Chaum, 90a] D. Chaum, A. Fiat, M. Naor, Untraceable electronic cash, Advances in Cryptology: CRYPTO ’88, 1990, 319–327.

[Chaum, 83] D. Chaum, Blind signatures for electronic payments, Advances in Cryptology: CRYPTO ’82, 1983, 199–203.

[Brands, 93] S. Brands, Untraceable off-line cash in wallet with observers (extended abstract), Advances in Cryptology: CRYPTO ’93, 1993.

[Okamoto and Ohta, 92] T. Okamoto, K. Ohta, Universal electronic cash, Advances in Cryptology: CRYPTO ’91, 1992, 324–337.

[Chaum, 90b] D. Chaum, Online cash checks. Advances in Cryptology: EUROCRYPT ’89, 1990, 288–293.

[Bellare, 00] M. Bellare, J. Garay, R. Hauser et al., Design, Implementation and Deployment of the iKP Secure Electronic Payment System, IEEE Journal on Selected Areas in Communications, 2000, 611-627.

[Ford and Baum, 97] W. Ford, M. Baum, Secure Electronic Commerce, Prentice Hall, 1997.

[Bella, 00a] G. Bella, F. Massacci, L. Paulson, P. Tramontano, Making sense of specifications: the formalization of SET (extended abstract), Proceedings of the 2000 Security Protocols Workshop, Lecture Notes in Comp. Sci., 2000, 74–81.

[Merkow, 98] S. Merkow, J. Breithhaupt, K. Wheeler, Building SET Applications for Secure Transactions, John Wiley & Sons, 1998.

[Jarupunphol and Mitchell, 03] P. Jarupunphol, C.J. Mitchell, Measuring 3-D Secure and 3D SET against e-commerce end-user requirements. Proceedings of the 8th Collaborative Electronic Commerce Technology and Research Conference, 2003, 51–64.

[Wrona, 01] K. Wrona, M. Schuba, G. Zavagli, Mobile payment — state of the art and open problems. In Fiege, L., Mühl, G., and Wilhelm, U. G., editors, Proceedings of 2nd International Workshop WELCOM, volume 2232 of Lecture Notes in Computer Science, Springer–Verlag, 2001, 88-100.

[Lawrence, 98] E. Lawrence, B. Corbitt, A. Tidwell, J. Fisher, et al., Internet Commerce: Digital Models for Business, John Wiley and Sons, 1998.

[Stallings, 95] W. Stallings, Network and Internetwork Security: Principles and Practice, Prentice Hall, Chapter 1, 1995.

[Bhimani, 96] A. Bhimani, Securing the Commercial Internet, Communications of the ACM, 39(6), 1996.

[Needham and Schroeder, 78] R. Needham, M. Schroeder, Using encryption for authentication in large networks of computers, CACM 21(12), 1978, 993-999.

[Lowe, 96] G. Lowe, Breaking and fixing the Needham-Schroeder public-key protocol using FDR, Tools and Algorithms for the Construction and Analysis of Systems, volume 1055 of Lecture Notes in Computer Science, 1996, 147-166.

[Dolev and Yao, 83] D. Dolev, A. Yao, On the security of public key protocols, IEEE TIT 29(2), 1983, 198-208.

[Abadi and Rogaway, 02] M. Abadi, P. Rogaway, Reconciling two views of cryptography (The computational soundness of formal encryption), Journal of Cryptology, 15(2), 2002, 103–127.

[Micciancio and Warinschi, 04] D. Micciancio, B. Warinschi, Soundness of Formal Encryption in the Presence of Active Adversaries, TCC, 2004.

[Durgin, 99] N. Durgin, P. Lincoln, J. Mitchell, A. Scedrov, Undecidability of Bounded Security Protocols, FLOC's Workshop on Formal Methods and Security Protocols, 1999.

[Mitchell, 97] J. Mitchell, M. Mitchell, U. Stern, Automated Analysis of Cryptographic Protocols using Murphi, IEEE Symposium on Security and Privacy, 1997, 141-154.

[Millen and Shmatikov, 01] J. Millen, V. Shmatikov, Constraint solving for bounded process cryptographic protocol analysis. Proc. 8th ACM Conference on Computer and Communications Security, 2001.

[Basin, 05] D. Basin, S. Mödersheim, L. Viganò, OFMC: A symbolic model checker for security protocols, International Journal of Information Security 4(3), 2005, 181-208.

[Paulson, 98] L.C. Paulson, The Inductive Approach to Verifying Cryptographic Protocols, Journal of Computer Security, 6(1), 1998, 85–128.

[Armando, 05] A. Armando et al., The Avispa Tool for the automated validation of internet security protocols and applications, Proceedings of CAV 2005, Computer Aided Verification, 2005.

[Von Oheimb, 05] D. Von Oheimb, The High-Level Protocol Specification Language HLPSL developed in the EU project AVISPA, Proceedings of APPSEM 2005 Workshop, 2005.

[Ryan, 00] P. Ryan, M. Goldsmith, G. Lowe, B. Roscoe, et al., Modelling & Analysis of Security Protocols, Addison Wesley, 2000.

[Armando and Compagna, 04] A. Armando, L. Compagna, SATMC: a SAT-based model checker for security protocols. In Proceedings of the 9th European Conference on Logics in Artificial Intelligence (JELIA’04), volume 3229 of LNAI, Lisbon, Portugal, Springer-Verlag, 2004, 730–733.

[Boichut, 05] Y. Boichut, P-C Heam, O. Kouchnarenko, Automatic Verification of Security Protocols Using Approximations. Research Report RR-5727, INRIA-Lorraine - CASSIS Project, 2005.

[Heather, 00] J. Heather, G. Lowe, S. Schneider, How to prevent type flaw attacks on security protocols, Proceedings of CSFW, 2000.

[Santiago and Vigneron, 05] J. Santiago, L. Vigneron, Study for Automatically Analysing Non-repudiation. In Actes du 1er Colloque sur les Risques et la Sécurité d'Internet et des Systèmes, CRiSIS, 2005, 157-171.

[Bella, 00b] G. Bella, F. Massacci, L.C. Paulson, P. Tramontano, Formal verification of cardholder registration in SET. In F. Cuppens, Y. Deswarte, D. Gollman, and M. Waidner, editors, Computer Security ESORICS 2000, LNCS 1895, 2000, 159-174.

[Bella, 02] G. Bella, F. Massacci, L.C. Paulson, The verification of an industrial payment protocol: the SET purchase phase. ACM Conference on Computer and Communications Security, 2002, 12-20.

[Fritscher and Kump, 00] M. Fritscher, O. Kump, Security And Productivity Improvements - Sufficient For The Success Of Secure Electronic Transaction? Proceedings of the 8th European Conference on Information Systems, 2000, 909-913.